Università degli Studi di Pisa
Dipartimento di Ingegneria dell'Informazione

Scuola di Ingegneria

Tesi di Laurea Magistrale in Ingegneria Robotica e dell'Automazione

Controllo ottimo di sciami di agenti mediante Descriptor Functions Framework e validazione sperimentale.

Candidato:
Alessandro Salvetti
Matricola 453264

Relatori:
Prof. Mario Innocenti
Prof. Lorenzo Pollini

Anno Accademico 2014–2015

La teoria è quando si sa tutto e niente funziona.
La pratica è quando tutto funziona e nessuno sa il perché.
Noi abbiamo messo insieme la teoria e la pratica:
non c'è niente che funzioni... e nessuno sa il perché!

— Albert Einstein

A mamma e babbo.

Indice

Elenco delle figure

Sommario

Lo sviluppo di sistemi *multi-agente*, riconfigurabili per l'esecuzione di task differenti, è un tema di particolare interesse all'interno della comunità scientifica. A partire dallo studio dei comportamenti animali, in cui la cooperazione fra i vari elementi del gruppo permette di raggiungere obbiettivi al di sopra delle capacità del singolo, sono state sviluppate molte teorie e tecniche che permettono di coordinare squadre di robot.

In questa tesi è stato utilizzato ed ampliato il *Framework delle Funzioni Descrittive* il quale permette il controllo di sciami di agenti eterogenei. La tecnica di controllo proposta, descrivendo analiticamente il sensore o il carico pagante montato su ogni agente, permette di generare traiettorie che ne ottimizzano l'uso, in funzione dell'obbiettivo del task. Il Framework è stato ampliato per garantire che vengano evitate collisioni con gli ostacoli, conservando tuttavia i criteri di ottimalità. Sono stati inoltre aggiunti controlli riguardati le distanze inter-agente così da mantenere gli agenti in gruppo evitando che si scontrino tra di loro. Il lavoro svolto dimostra anche come il Framework permetta la ricerca e l'inseguimento di intrusi all'interno dell'ambiente di lavoro.

È stato infine sviluppato un testbed per validare sperimentalmente i risultati teorici. Lo sciame è composto da veicoli terrestri autonomi la cui posizione, velocità e orientamento sono ottenuti tramite sensore GPS e magnetometro. Lo scambio di informazioni fra i veicoli avviene tramite rete wireless. L'accesso al mezzo sfrutta un protocollo di tipo *Time Division Multiple Access*, sincronizzato sul clock del sistema GPS.

Abstract

The development of multi-agent systems, which are able perform different tasks, is a topic of particular interest within the scientific community. In nature, there are many living species that join into groups in order to achieve goals beyond the capacity of a single element. Starting from those behaviors, researchers develop many control techniques that coordinates teams of robots.

This thesis uses the *Descriptor Functions Framework* in order to control swarms of heterogeneous agents. The Framework analytically describes the payload mounted on agents, for which reason it can move the swarm optimizing the sensing performances. The Framework is then expanded in order to ensure obstacle avoidance, maintaining optimality conditions. In addition, another control technique, that ensure inter-agent distance constraints, has been developed. Furthermore, this thesis demonstrates that the *Descriptor Functions Framework* can be used to track and search possible intruders.

In order to prove the theoretical results, an hardware testbed has been developed. The controlled swarm is made up of autonomous land vehicles whose position, speed and orientation are obtained by a GPS sensor and a magnetometer. Communication between agents is wireless.A Time Division Multiple Access protocol synchronized with the clock of the GPS system guarantees the absence of collisions in the information exchange process.

Ogni nostra cognizione
principia dai sentimenti.

— Leonardo da Vinci

Ringraziamenti

Desidero ringraziare tutti coloro che mi sono stati accanto durante il periodo di stesura della tesi e non, a chi mi ha dato consigli, posto critiche e fatto osservazioni. Ringrazio innanzitutto il Professor Mario Innocenti, relatore, per avermi permesso di lavorare in assoluta libertà sugli argomenti che ho trovato di maggior interesse. Inoltre Lo ringrazio per gli spunti, le osservazioni e gli scambi di opinioni portati avanti durante tutto il corso della tesi. Ringrazio inoltre il secondo relatore, il Professor Lorenzo Pollini, per avermi aiutato nello sviluppo dell'hardware e del software e il Dottorando Giovanni Franzini con il quale ho condiviso cocenti giornate di test e tante risate.

Ringrazio poi tutti gli amici e compagni dell'Università con cui ho condiviso fatiche, progetti, opinioni, pasti, arrabbiature e momenti di puro divertimento. Un grazie va quindi a Filippo Fabiani, Marco Tranzatto, Alessandro Tondo, Anna Mannucci, Simone Della Tommasina, Paolo Pazzaglia, Roger Nuti e tutti gli altri compagni di corso. Ringrazio Giulia per avermi supportato e sopportato in tutti questi anni e per essere stata sempre presente. Come non ringraziare poi tutti i "fanti di Marina": Lorenzo, Micheal, Davide, Giovanni, Matteo e Alessia con i quali ho trascorso giornate memorabili e con i quali spero di condividere ancora tante esperienze. Un ricordo va anche alla San Diego State University e ai ragazzi conosciuti laggiù.

Infine, il ringraziamento principale ai miei genitori: grazie di tutto.

Pisa, Luglio 2015 A. S.

Introduzione

Negli ultimi anni si è assistito a una crescente attività di ricerca sul controllo cooperativo e sul moto coordinato dei sistemi multi agente (*MAS, Multi Agent Systems*). Questi ultimi si compongono di veicoli, robot, reti di sensori e in generale anche di elementi eterogenei appartenenti a una rete o infrastruttura comune.

I forti progressi tecnologici nell'ambito della miniaturizzazione dei sistemi elettromeccanici, dello sviluppo di sensori sempre più accurati e del networking, hanno reso applicabili studi che prima erano soltanto teorici e hanno dato un forte impulso alla ricerca in questi campi.

Le prime analisi, a livello ingegneristico, hanno visto la nascita di modelli e sistemi matematici atti a riprodurre, il più fedelmente possibile, situazioni già esistenti in natura. Si pensi ad esempio ai branchi di animali, agli sciami di insetti o agli stormi di uccelli: la cooperazione e la sincronizzazione di tutti gli elementi porta a risultati superiori rispetto a quelli attuabili dal singolo. In questo ambito ormai la letteratura è abbondante e le soluzioni apportate sono molteplici. Per approfondimenti sul confronto tra robotica e biologia si vedano [Bonabeau et al., 1999], [Kennedy et al., 2001] e [Passino, 2005].

Le applicazioni dei *MAS* sono molteplici e appartenenti ad ambiti diversi: dal monitoraggio dei fenomeni naturali, alla ricerca di obbiettivi in ambienti pericolosi, passando per la sorveglianza e il pattugliamento (si veda [LaValle, 2006], [Hussein, Stipanović et al., 2007a], [Bullo, Cortés et al., 2009], [Mesbahi e Egerstedt, 2010]). Gli agenti dei *MAS* devono essere in grado di operare in concerto l'uno con l'altro per conseguire gli obiettivi a livello di sistema, pur avendo un accesso limitato alle risorse computazionali, alle comunicazioni e alla capacità di rilevamento. Così facendo si possono completare task difficili, se non impossibili, per il singolo agente. Inoltre un gruppo di agenti, rispetto al singolo, fornisce intrinsecamente robustezza ai *single point of failure* e ai guasti sui canali di comunicazione.

Gli argomenti trattati in questa tesi riguardano i sistemi multi agente. In particolare lo sciame di agenti è controllato attraverso il Framework delle Funzioni Descrittive

in grado di gestire con semplicità molti compiti diversi quali self-deployment, ricerca e pattugliamento fino al tracking di intrusi. Il Framework è stato sviluppato da Niccolini, Innocenti et al. e ampliato da Ferrari Braga et al.

Inoltre per validare sperimentalmente il lavoro teorico svolto è stato sviluppato un testbed composto da veicoli autonomi che simulano lo sciame di agenti.

Il primo capitolo offre una descrizione dettagliata delle caratteristiche e del funzionamento del Framework delle Funzioni Descrittive.

Il secondo capitolo, invece, mostra come si possa ampliare il Framework in modo che gli agenti evitino eventuali ostacoli presenti nell'area di lavoro.

Nei capitoli 3 e **4** vengono introdotti vincoli sulle distanze massime e minime che gli agenti possono avere fra di loro. Nel Capitolo 3 l'estensione si affianca al Framework, mente nel Capitolo 4 viene direttamente inclusa al suo interno.

Il quinto capitolo introduce un nuovo task per lo sciame di agenti: la ricerca e l'inseguimento di eventuali intrusi nell'area operativa.

Il sesto capitolo, invece, volge l'attenzione verso la dinamica a singolo integratore con cui sono modellati gli agenti. Si dimostra quindi come il Framework possa lavorare anche con dinamiche del secondo ordine.

Il settimo capitolo, infine, espone come è stato realizzato il testbed di validazione dell'apparato teorico sviluppato e mostra i risultati ottenuti.

Capitolo 1

Descriptor Functions Framework

Il Framework delle Funzioni Descrittive rappresenta una metodologia per il controllo di uno sciame di agenti eterogenei. Ogni agente, sia esso un veicolo terrestre, marino o aereo, viene descritto matematicamente in base alle capacità del sensore di payload che monta. Da questa rappresentazione, e da quella del task da portare a termine, viene generata una legge di controllo sub-ottima che muove gli agenti nello spazio, portandoli a compiere l'obbiettivo richiesto.

Il Framework è stato concepito e analizzato da Niccolini [Niccolini, Innocenti et al., 2010] [Niccolini, 2011] [Niccolini, Pollini et al., 2014] ed è stato rivisto ed esteso da Ferrari Braga [Ferrari Braga et al., s. d.] [Ferrari Braga, 2013]. In questa tesi si analizza nuovamente il Framework sotto nuovi punti di vista e se ne estendono le potenzialità e gli usi.

1.1 Modellazione matematica degli agenti

Come già accennato precedentemente, gli agenti dello sciame sono rappresentati mediante funzioni matematiche dette appunto Funzioni Descrittive o ADF (agent Descriptor Functions) per brevità. Queste funzioni cercano di riprodurre non tanto la forma fisica dell'agente, che anzi viene considerato puntiforme, quanto piuttosto la capacità di acquisire (o portare) informazioni. In sostanza la ADF cerca di descrivere, in qualche modo, quelle che sono le proprietà del sensore o del carico di payload dell'agente.

Si consideri, ad esempio, un robot mobile che debba perlustrare una determinata area. Probabilmente esso monterà a bordo una telecamera con un certo angolo di vista (Field of View, o FoV), che permette al robot di riconoscere determinati obbiettivi o intrusi nell'ambiente esplorato. In questo caso, la ADF dell'agente sarà una funzione che rispecchia le caratteristiche della telecamera. Dovrà essere ad esempio nulla all'esterno di un certo angolo, che corrisponde al FoV, così da indicare

che il sensore non acquisisce informazioni fuori dal suo cono di vista. Dovrà inoltre avere valori tanto più piccoli all'aumentare della distanza dall'agente puntiforme così per rispecchiare la caratteristica della telecamera di avere un fuoco limitato e vicino all'agente, e quindi di saper riconoscere meglio oggetti vicini piuttosto che lontani.

Un altro esempio classico prevede da uno sciame d'agenti che si deve disporre per portare copertura wifi nella più vasta area possibile. Il payload dell'agente questa volta non è costituito da un sensore per acquisire informazioni, ma da un carico che porta informazioni, che aggiunge qualcosa all'ambiente. A differenza del caso precedente questa volta la Funzione Descrittiva sarà omnidirezionale, con un picco nel punto in cui si trova l'antenna del ripetitore, e con valori via via decrescenti man mano che ci si allontana dall'agente.

Dopo questi brevi esempi risulta semplice porre delle linee guida su come devono essere costruite le Funzioni Descrittive. Tali funzioni dovranno avere valori elevati in quelle zone per cui il sensore ha una maggiore precisione e accuratezza o dove il carico riesce a portare maggiore informazione. Viceversa avranno valori inferiori o nulli laddove il sensore ha delle carenze, come Field of View, direzioni preferenziali, obbiettivi troppo lontani ecc... Non hanno ovviamente senso valori negativi delle ADF.

Attualmente esistono vari tipi di sensori o carichi paganti adatti a tutte le possibili missioni che si richiede di svolgere ad uno sciame di agenti. Tuttavia si possono in generale dividere in due grandi categorie, quelli che acquisiscono o forniscono informazioni omnidirezionalmente e quelli che lo fanno in una specifica direzione e che quindi posseggono un determinato campo di vista.

Le ADF che nel seguito si andranno ad analizzare sono proprio divise in queste due categorie, funzioni *omnidirezionali* e funzioni con *Field of View*.

Si vuole infine richiamare l'attenzione sulla proprietà di eterogeneità che il Framework prevede per lo sciame. In esso infatti non solo possono coesistere veicoli diversi, ma ogni agente può montare un payload differente dagli altri. Perciò lo sciame può essere rappresentato da agenti con ADF diversa, non c'è alcuna necessità che debbano essere tutte uguali. Ciò che gli agenti devono invece condividere è il compito: non è ammesso che nello stesso sciame coesistano agenti che portano informazioni con altri che ne acquisiscono.

Un ultimo particolare tecnico: come sarà più chiaro in seguito la Funzione Descrittiva dell'agente deve essere differenziabile con continuità rispetto alla posizione dell'agente stesso.

1.1.1 Funzioni Descrittive omnidirezionali

La prima Funzione Descrittiva sviluppata ([Niccolini, Innocenti et al., 2010]) è costituita dalla curva gaussiana (Figura 1.1). Essa ben si presta a rappresentare tutta

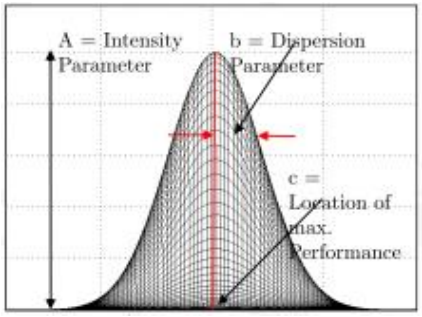


Figura 1.1: Funzione gaussiana.

quella gamma di sensori omnidirezionali con prestazioni elevate vicino al sensore stesso che poi degradano rapidamente all'aumentare della distanza. Un esempio pratico sono le antenne per comunicazioni radio: la potenza del segnale emesso degrada proprio come un esponenziale. La funzione gaussiana possiede anche la proprietà di essere radialmente simmetrica, qualità molto comune fra i carichi di payload. Tale caratteristica conduce tra l'altro ad interessanti conclusioni riguardo al significato del controllo generato dal DF Framework, per maggiori dettagli si veda [Niccolini, 2011].

La funzione mostrata in Figura 1.1 è tuttavia valida solo per agenti che si debbano muovere lungo una dimensione. Per portare l'agente a muoversi sul piano o nello spazio 3D è necessario passare ad una funzione che abbia dominio in $\mathbb{R}^2$ o in $\mathbb{R}^3$.

In generale si può scrivere:

$$\mathrm{ADF}_{gaussiana} = Ae^{-\frac{1}{2}(q-p_i)^T S^{-1}(q-p_i)}$$

dove A è l'ampiezza massima della funzione; $p \in \mathbb{R}^n$ è la posizione dell'agente nello spazio n dimensionale (nei casi pratici $n = 2, 3$); $S = diag(\sigma_1, \sigma_2, ..., \sigma_n)$ è la matrice di dispersione della gaussiana; $q \in \mathbb{R}^n$ è lo spazio in cui si muove l'agente.

Se i valori della matrice di dispersione S sono tutti uguali la Funzione Descrittiva risulta radialmente simmetrica, altrimenti si ottiene una superficie che decresce con velocità differenti lungo le direzioni dello spazio che si sta considerando.

Questa possibilità di manipolazione della forma della funzione permette di adattarla facilmente alle più disparate situazioni. Si pensi ad esempio ad un veicolo dotato di sensor ring (un anello di sensori posizionati tutti attorno alla geometria dell'agente), modificando accuratamente i valori di dispersione si può ottenere una valida rappresentazione dell'agente (o meglio del suo sensore) nello spazio. Un esempio di ciò che si può ottenere modificando ampiezza, dispersione e punto dove si trova l'agente è rappresentato in Figura 1.2.

Un'altra possibile funzione adatta al caso di payload con propagazione omnidirezionale è stata sviluppata in [Hussein, Stipanović et al., 2006] anche se nell'articolo

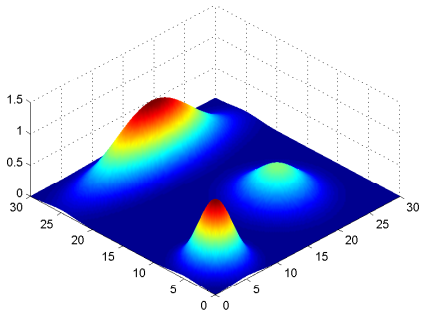


Figura 1.2: Rappresentazione di tre agenti con funzione gaussiana ma con ampiezze e dispersioni differenti.

in questione, e nei successivi dell'autore, tale funzione non è concepita per il DF Framework, ma per un'altra tecnica simile.

L'equazione in questo caso risulta essere polinomiale e anch'essa fa propria la caratteristica di essere a simmetria radiale. Differisce dalla funzione gaussiana per il fatto di essere matematicamente limitata, ossia il suo valore è esattamente nullo oltre una determinata distanza r dall'agente. Questa proprietà descrive con maggiore accuratezza il comportamento di sensori, tuttavia ingegneristicamente parlando si può considerare nulla anche la curva gaussiana per valori superiori a 5σ.

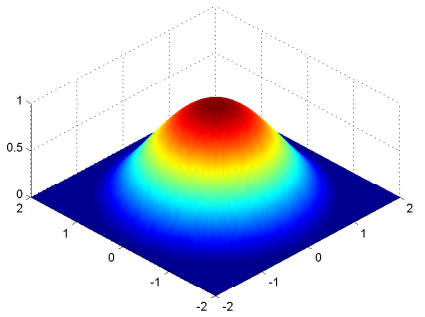


Figura 1.3: Rappresentazione di un agente dotato di ADF polinomiale con $A = 1$ e $r = 2$.

L'equazione di questa ADF risulta essere:

$$\text{ADF}_{polinomiale} = \begin{cases} \frac{A}{r^4}(\|(q - p_i)\|^2 - r^2)^2, & \text{se } \|(q - p_i)\|^2 \leq r^2 \\ 0, & \text{altrimenti} \end{cases}$$

Anche in questo caso A è l'ampiezza massima della funzione che si ottiene per $q = p_i$. Un difetto di questa ADF è il fatto di non poter essere "allungata" in una direzione, come invece può fare la gaussiana. In Figura 1.3 è possibile osservare la forma risultante.

Infine, con l'obbiettivo di ampliare ancora la gamma delle possibili agent Descriptor Functions, è stata sviluppata una funzione sinusoidale che modella anch'essa payload omnidirezionali. L'equazione della curva è la seguente:

$$\mathrm{ADF}_{sinusoidale} = \begin{cases} A\cos^2\left(\frac{\pi(p_1-q_1)}{2s_1}\right)\cos^2\left(\frac{\pi(p_2-q_2)}{2s_2}\right) & \text{per } \frac{(p_1-q_1)^2}{s_1^2} \leq 1 \wedge \frac{(p_2-q_2)^2}{s_2^2} \leq 1 \\ 0, & \text{altrimenti} \end{cases}$$

in questo caso sono state messe in evidenza le componenti del vettore p_i e di q per semplicità. Esattamente come nel caso della ADF gaussiana, i parametri di dispersione s_1 e s_2 permettono di allungare la funzione in una direzione piuttosto che in un'altra. Invece, similmente alla funzione polinomiale, risulta essere limitata nello spazio, mantenendo la proprietà di differenziabilità. In Figura 1.4 è possibile notare la somiglianza con la funzione gaussiana e in particolare con la Figura 1.2.

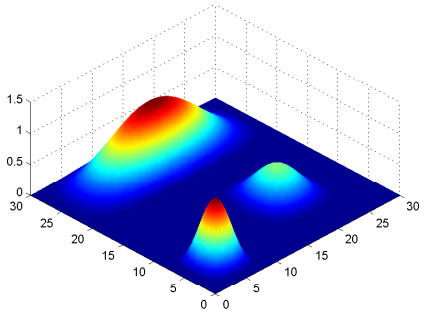
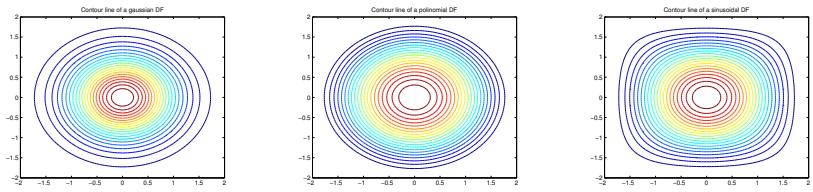


Figura 1.4: Rappresentazione di tre agenti con funzione sinusoidale ma con ampiezze e dispersioni differenti.

In verità questa funzione costruita con l'utilizzo del coseno perde la proprietà di simmetria radiale. Le curve di livello mostrate in Figura 1.5 rendono bene l'idea della differenza. Mentre le altre ADF generano curve circolari, la DF sinusoidale descrive curve più "quadrate", che si possono ben adattare a veicoli terrestri dotati di sensor ring.



(a) $ADF_{gaussiana}$ **(b)** $ADF_{polinomiale}$ **(c)** $ADF_{sinusoidale}$

Figura 1.5: Curve di livello di diverse Funzioni Descrittive omnidirezionali.

1.1.2 Funzioni Descrittive con Field of View

Finora sono state trattate soltanto funzioni con valori positivi tutto intorno all'agente tuttavia un'ampia categoria di sensori ha un ben determinato cono di funzionamento. Un sensore ad ultrasuoni, una telecamera o un sensore ad infrarossi necessitano di una ADF che descriva proprio questo comportamento.

In [Ferrari Braga, 2013] e [Ferrari Braga et al., s. d.] si affronta proprio questo problema. La soluzione proposta rende molto semplice la costruzione di Funzioni Descrittive con campo di vista a partire da quelle omnidirezionali. Inizialmente si genera una funzione, detta di Field of View, nulla nello spazio tranne per un certo angolo o cono, come mostrato in Figura 1.7.

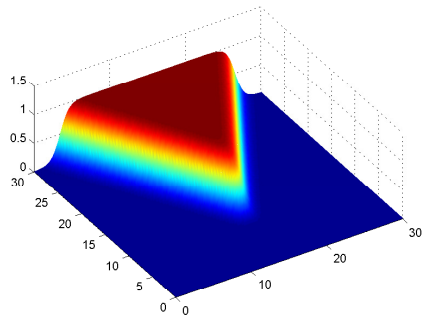


Figura 1.6: Funzione Field of View.

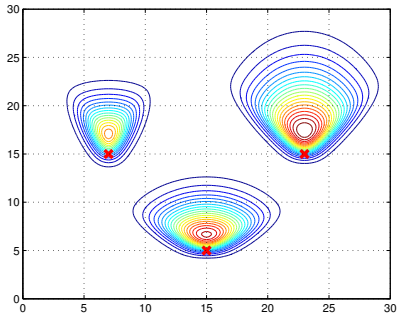
Tale funzione viene generata dall'intersezione di due semipiani, costruiti attraverso la generalizzazione delle funzioni sigmoidi nello spazio. Matematicamente si ha:

$$FoV = SP_r(\phi, p_i, q) SP_l(\phi, p_i, q)$$
$$SP_r(\phi, p_i, q) = \left(1 + e^{-K[(q_1 - p_1)\cos(\phi) + (q_2 - p_2)\sin(\phi)]}\right)^{-1}$$
$$SP_l(\phi, p_i, q) = \left(1 + e^{-K[-(q_1 - p_1)\cos(\phi) + (q_2 - p_2)\sin(\phi)]}\right)^{-1}$$

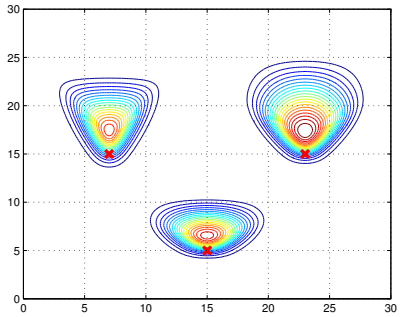
Ad una prima occhiata l'espressione può risultare complessa ma in verità è molto semplice. SP_r e SP_l sono funzioni sigmoidi che danno luogo ad una specie di gradino. Il parametro K ne aumenta o riduce la ripidità. I pedici r ed l invece denotano in quale direzione è orientato il gradino, se a destra o a sinistra, a meno di una rotazione dell'angolo ϕ. Moltiplicando questi due "semipiani" si ottiene la funzione FoV.

Lo stesso procedimento può essere eseguito con curve simili alle sigmoidi come ad esempio l'arcotangente, opportunamente scalata e traslata.

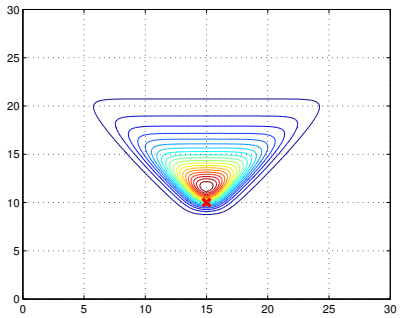
Giunti a questo punto costruire Funzioni Descrittive con campo di vista risulta davvero semplice: è sufficiente moltiplicare la ADF omnidirezionale con la funzione FoV. Otterremo così funzioni gaussiane, polinomiali, o sinusoidali con Field of View, mostrate in Figura 1.7). Si può notare come le caratteristiche di tali funzioni siano



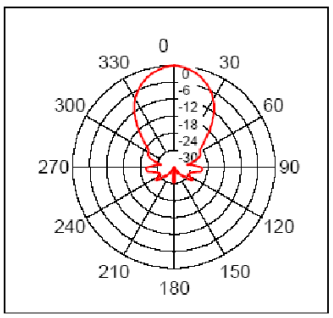
(a) $ADF_{gaussiana}$ *con FoV*



(b) $ADF_{sinusoidale}$ *con FoV*



(c) $ADF_{sigmoide}$ *con FoV*



(d) *Beam pattern di un sensore ad ultrasuoni.*

Figura 1.7: Curve di livello di diverse Funzioni Descrittive dotate di Field of View (a diversi gradi di ampiezza). La X rossa indica la posizione dell'agente. L'ultima immagine mostra il beam pattern di un sensore ad ultrasuoni, si nota come le ADF di Figura 1.7a e 1.7b ben rappresentano questo pattern.

mantenute nella parte non nulla del FoV. Ad esempio le ADF gaussiane con campo di vista risultano ancora matematicamente illimitate e più tondeggianti rispetto a quelle sinusoidali. Ovviamente viene persa la simmetria radiale.

Inoltre è interessante notare come il massimo di queste funzioni non sia in corrispondenza dell'agente ma in posizione avanzata. Questo particolare ben rispecchia le proprietà di molti sensori di possedere maggiore accuratezza ad una breve distanza dalla zona di applicazione, si pensi ad esempio al fuoco di una telecamera.

Per capire invece quanto bene le funzioni sviluppate rappresentino i sensori reali si può far riferimento alla Figura 1.7d. Essa rappresenta il beam pattern di un sensore ad ultrasuoni commerciale. Si può notare come la seconda gaussiana con FoV di Figura 1.7a ben si adatti al compito di modellare questo dispositivo.

È stata infine sviluppata, sempre in [Ferrari Braga et al., s. d.], un'altra ADF con campo di vista. In questo caso la funzione FoV viene moltiplicata per un'altra sigmoide. Si ottiene così una funzione più spigolosa delle precedenti come si vede in

Figura 1.7c. L'equazione di tale curva è omessa per brevità.

1.1.3 Orientazione delle Funzioni Descrittive

Tutte le Funzioni Descrittive proposte finora risultano muoversi assieme all'agente in quanto la posizione p_i fa sempre parte dell'equazione caratteristica delle ADF. Tuttavia non è possibile orientarle a piacimento, in altre parole, all'agente non è permesso ruotare. Questo inconveniente è stato risolto in [Ferrari Braga, 2013] andando a dividere il sistema di riferimento inerziale Q (fisso, che non può ruotare) da quello dell'agente B (a cui ora è permesso di cambiare orientazione).

Così facendo le equazioni delle ADF descritte nei precedenti paragrafi risultano appartenere al frame body dell'agente. Risulta perciò necessario applicare una rotazione che le porti dal frame body a quello inerziale. Ciò si ottiene attraverso ${}^{Q}C_{B}$, ossia la matrice di rotazione attorno all'asse $\hat{z}$ che introduce la rotazione di un angolo θ :

$${}^{Q}C_{B} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & -\cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Nel caso bidimensionale si prendono soltanto le prime due righe e colonne di ${}^{Q}C_{B}$. Per ruotare le ADF dal frame body a quello inerziale è quindi necessario eseguire la seguente operazione:

$${}^{Q}(q - p_i) = {}^{Q}C_{B} \cdot {}^{B}(q - p_i)$$

che significa semplicemente andare ad operare la seguente sostituzione in tutte le Funzioni Descrittive:

$$\begin{cases} (q_1 - p_1) \longrightarrow +(q_1 - p_1)\cos(\theta) + (q_2 - p_2)\sin(\theta) \\ (q_2 - p_2) \longrightarrow -(q_1 - p_1)\sin(\theta) + (q_2 - p_2)\cos(\theta) \end{cases}$$

La tecnica appena svolta è applicabile a tutte le ADF illustrate finora, anche se ovviamente, non porta alcun vantaggio nel caso si applichino a funzioni con simmetria radiale. In Figura 1.8 è possibile osservare alcune Funzioni Descrittive ruotate di 45 gradi. Si è ottenuto quindi un agente in grado di spostarsi e ruotare nello piano.

1.2 Modellazione matematica dei task

I task che lo sciame deve portare a termine sono descritti, all'interno del DF Framework, come funzioni dello spazio Q. Il concetto di base è lo stesso di quello utilizzato per modellare le ADF. In questo caso la Task Descriptor Function (nel seguito TDF) avrà valori tanto più elevati tanto più richiede risorse da parte dello sciame. Possono esistere perciò porzioni dello spazio Q in cui è meno importante

la presenza degli agenti (TDF più bassa) e altre in cui lo è di più (TDF più alta). Descrive in sostanza come devono essere collocate le risorse (gli agenti) all'interno dello spazio Q. Da qui in avanti la Funzione Descrittiva del Task verrà rappresentata matematicamente come

$$d^*(q,t) : Q \longrightarrow \mathbb{R}^+$$

Come per le ADF anche le Funzioni dei task devono essere differenziabili con continuità.

1.2.1 Current Task Descriptor Function

Prima di mostrare alcuni esempi di modellazione dei task, è necessario introdurre il concetto di Current Task Descriptor Function (o cTDF). Essa è definita a partire dalle ADF di singoli agenti che d'ora in poi verranno indicate come:

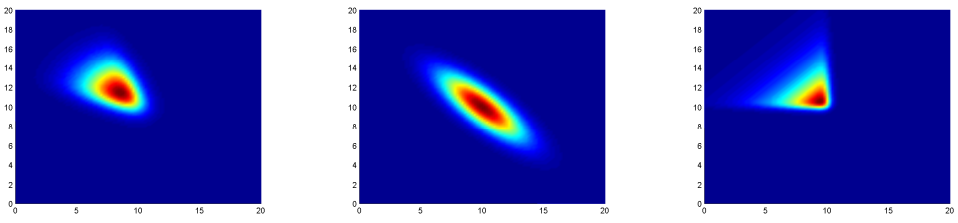
$$d_i(p_i,q) : Q \longrightarrow \mathbb{R}^+$$

La cTDF è definita come la somma di tutte le Funzioni Descrittive degli agenti e indica come le risorse (rappresentate dal payload degli agenti) sono distribuite nello spazio Q:

$$D_i(p,q) = \sum_{i=1}^{N_a} d_i(p_i,q)$$

dove N_a è il numero degli agenti e $p = [p_1^T, p_2^T, ..., p_{N_a}^T]^T$.

Essa rappresenta una sorta di Funzione Descrittiva per l'intero sciame. Tanto più la cTDF è simile alla TDF tanto meglio viene eseguito il task. In Figura 1.9 è rappresentato quello che dovrebbe essere il comportamento base dello sciame: gli agenti, raggruppandosi al centro dell'ambiente, danno luogo ad una cTDF il più possibile simile, come forma e posizione, alla TDF.



(a) $ADF_{gaussiana\ FoV}$ **(b)** $ADF_{gaussiana}$ **(c)** $ADF_{sigmoide\ FoV}$

Figura 1.8: Varie Funzioni Descrittive ruotate di 45 gradi.

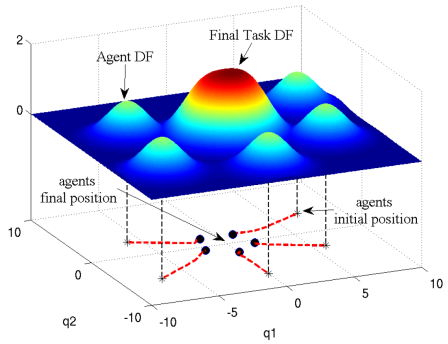


Figura 1.9: Esempio di come agisce il DF Framework.

1.2.2 Uniform Deployment Task

La forma più semplice di TDF è rappresentata da una costante su tutto l'ambiente Q. Avremo perciò:

$$d^*(q,t) = const \tag{1.1}$$

Il risultato che si ottiene vede gli agenti spostarsi in modo da disporsi il più uniformemente possibile sull'ambiente, compatibilmente con le loro ADF. In Figura 1.10 è possibile vedere il movimento dello sciame nel complesso e la sua disposizione finale. In particolar modo in Figura 1.10b è rappresentata la cTDF sull'ambiente Q.

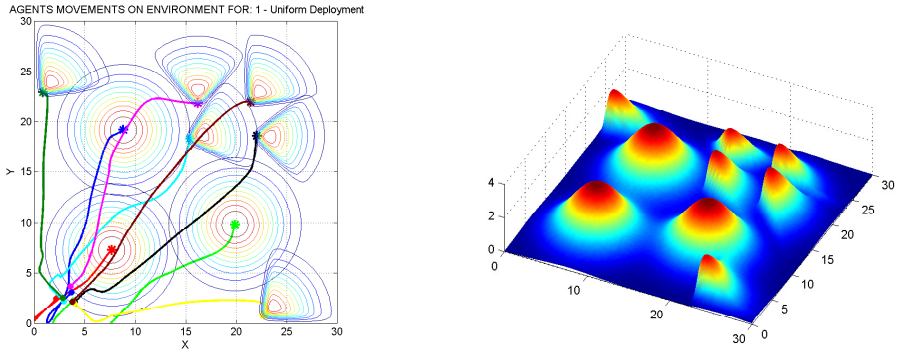
Lo uniform deployment task è stato affrontato in moltissimi modi in letteratura, passando dai diagrammi di Voronoi (si veda [Bullo, Cortés et al., 2009]) alla teoria dei campi potenziali ([Howard et al., 2002]), fino agli algoritmi ad-hoc ([Meguerdichian et al., 2001]).

L'obbiettivo di questo task infatti non è per nulla banale e possiede parecchi campi di applicazione, si pensi ad esempio al caso discusso in Sezione 1.1 in cui agli agenti è richiesto di disporsi sul territorio in modo da massimizzare l'area di copertura di una ipotetica rete wireless.

1.2.3 Static Coverage Task

Questo task richiede che gli agenti si dispongano in una particolare area di interesse piuttosto che su tutto l'ambiente. Per fare un esempio potremmo considerare uno sciame di agenti a cui è richiesto di monitorare un certo evento accaduto in un'area specifica dell'ambiente su cui normalmente opera. A livello matematico la TDF risulta ancora una funzione costante nel tempo e nello spazio, ma con valori differenti a seconda del punto $q \in Q$ considerato:

$$d^*(q,t) = \bar{d}^*(q) \tag{1.2}$$



(a) *Percorso effettuato* **(b)** *cTDF all'istante finale*

Figura 1.10: Esempio di Uniform Deployment Task con nove agenti con ADF gaussiana di cui sei con FoV. Si può notare come alla fine lo sciame si sia disposto in modo da coprire il più possibile la superficie di interesse.

La Figura 1.9 mostra proprio task: si ha una TDF gaussiana (ma potrebbe avere una qualsiasi forma) al centro dell'ambiente e gli agenti che si spostano concentrandosi su di essa.

Anche questo problema di copertura di un'area specifica è stato affrontato da diversi autori in diversi modi. Di particolare rilevanza è il lavoro svolto da Bullo, Cortés e Martinez, in [Bullo, Cortés et al., 2009], in cui vengono ampiamente utilizzati i diagrammi di Voronoi . In [Schwager et al., 2007] e [Sharifi et al., 2015] si possono trovare delle variazioni sul tema con l'aggiunta di controlli adattivi o funzionali di costo.

1.2.4 Effective Coverage Task

L'effective coverage richiede agli agenti di eseguire una ricerca esaustiva sull'ambiente in cui si opera. In pratica si desidera che lo sciame si muova il più possibile sul territorio.

Questo task modella scenari in cui si desidera cercare qualcuno o qualcosa. Ad esempio potrebbe trattarsi della ricerca di superstiti in zone colpite da catastrofi, oppure di eventuali criticità all'interno di una zona. È pure possibile pensare di unire a questo task il precedente: una volta che gli agenti hanno individuato un dato obbiettivo il task muta a favore dello static coverage che porta lo sciame a radunarsi proprio attorno all'obbiettivo.

Questo compito è stato formulato da Hussein [Hussein, Stipanović et al., 2006] ed ampliato più volte fino all'aggiunta di un secondo sciame che coordini il primo [Hussein, Stipanović, Wang et al., 2007].

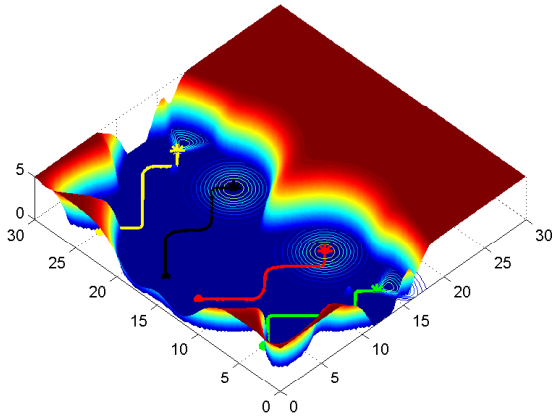


Figura 1.11: Esempio di Effective Coverage Task (ancora in corso).

Matematicamente, all'interno del DF Framework, esse viene rappresentato come segue:

$$d^*(q,t) = \max\left(0, \bar{C} - \int_0^t D(p(\tau), q)\,\mathrm{d}\tau\right) \tag{1.3}$$

Apparentemente complicata, l'equazione dell'effective coverage è invece abbastanza semplice: $\bar{C}$ indica quanta risorsa, da parte degli agenti, è necessaria affinché il territorio venga considerato sufficientemente controllato. Man mano che gli agenti si spostano il valore di d^* deve risultare piccolo se non nullo proprio nelle zone in cui si è già perlustrato ad indicare lo scarso interesse a osservare nuovamente tale luogo. Questo comportamento è modellato dall'integrale della cTDF.

In altre parole è come se gli agenti "mangiassero" la TDF, che all'inizio ha valore costante $\bar{C}$. In Figura 1.11 si può osservare quattro agenti nell'atto di eseguire la ricerca. Il valore della TDF lungo il percorso dello sciame è effettivamente nullo, ad indicare che la zona è già stata perlustrata. Il resto della funzione task ha invece ancora il valore nominale $\bar{C} = 5$ perché deve ancora essere raggiunta. Una volta che lo sciame avrà coperto tutto l'ambiente il task terminerà con valore ottimale della TDF pari a 0 (valore che non è detto che venga raggiunto).

1.2.5 Dynamic Coverage Task

L'obbiettivo del task prevede che lo sciame pattugli l'ambiente Q. Mentre per l'effective coverage era sufficiente che gli agenti perlustrassero una sola volta l'ambiente, adesso è richiesto che le porzioni di spazio precedentemente visitate vengano ricontrollate ancora e ancora.

In pratica si tratta di mantenere aggiornate le informazioni sul territorio. Questa volta, la funzione obbiettivo d^* non deve restare bassa dove sono già passati gli agenti,

ma deve pian piano tornare a crescere ad indicare la necessità di raccogliere o portare nuovamente informazioni in quella data area.

Questo processo, chiamato di decadimento dell'informazione, ad indicare la perdita di informazioni (in quanto datate) riguardanti il territorio, è modellabile analiticamente attraverso la seguente equazione alle derivate parziali:

$$\frac{\partial}{\partial t} I(q,t) = \delta I(q,t) + D(p,q)$$

con $\delta \leq 0$ costante che indica quanto velocemente l'informazione sul territorio decade (se $\delta = 0$ il task ricade nell'effective coverage). Quindi la TDF può essere descritta come segue:

$$d^*(q,t) = \max\left(0, \bar{C} - I(q,t)\right)$$

oppure, similmente al task di effective coverage, come:

$$d^*(q,t) = \max\left(0, \bar{C} + H(q,t) - \int_0^t D(p(\tau), q)\, \mathrm{d}\tau\right) \tag{1.4}$$

dove il termine $H(q,t)$ è il fattore che incrementa nel tempo il valore di d^*, così da richiedere nuovamente il passaggio degli agenti nelle zone visitate meno di recente. Ovviamente per far si che questo avvenga dovrà essere verificato che $\dot{H} > 0$.

1.2.6 Target Assignment Task

Si consideri lo scenario in cui ci siano più locazioni, all'interno dell'ambiente, che necessitino la presenza degli agenti. In questo caso quello che si desidera è che ci sia almeno un agente per obbiettivo, ammesso che il numero degli agenti lo consenta.

Detta $d_k^* = const$ la funzione che modella la richiesta di risorse del k-esimo obbiettivo allora la TDF complessiva può essere scritta come segue:

$$d^*(q,t) = \sum_{k=1}^{N_t} d_k^*(q) \tag{1.5}$$

In pratica è come avere più Static Coverage Task in un solo task.

Purtroppo il DF Framework da solo non basta a garantire che per ogni task ci sia almeno un agente. Tuttavia è possibile implementare degli interessanti algoritmi di assegnamento delle risorse, si veda [Ferrari Braga, 2013], che, individuato l'agente più adatto per tutti gli N_t target, fa in modo di portare, sempre con la legge di controllo del Framework, gli agenti al relativo task.

1.3 Controllo

Finora sono stati presentati i modi con cui il DF Framework descrive gli agenti e i task che questi devono eseguire. In Sezione 1.2.1 è stato esposto il concetto che

sta dietro al DF Framework: gli agenti si muovono verso zone in cui la TDF è più alta, cercando di rendere il più simile possibile la cTDF alla TDF, oppure di portare quest'ultima a zero.

In questa Sezione si discuterà di come, e secondo quali criteri, venga generato il controllo che muove gli agenti, e di quale sia lo loro dinamica.

1.3.1 Task Error Function

La Task Error Function, abbreviata TEF, è definita come:

$$e(q,p,t) = d^*(q,t) - D(p,q) = d^*(q,t) - \sum_{i=1}^{N_a} d_i(q,p_i)$$

e indica la discrepanza, in termini spaziali, fra la Funzione Descrittiva del Task, e quella dello sciame.

Quindi, per le affermazioni precedenti, la TEF è un indice di quanto il task sia ben eseguito. Più $e(q,p,t) \longrightarrow 0$ più l'obbiettivo viene raggiunto.

È doveroso sottolineare tuttavia che può sussistere il caso in cui $d^*(q,t) > D(q,p), \quad \forall q,p,t$. In questo caso la funzione d'errore non potrebbe mai raggiungere il valore nullo. Tuttavia questo inconveniente non è significativo, risulta essere un banale problema di scalatura delle funzioni, oppure indica che lo sciame, anche al meglio delle sue capacità non riesce a ricoprire per intero la richiesta di risorse del task. In ogni caso, l'importante è riuscire a minimizzare il più possibile la TEF, ossia portare lo sciame nella configurazione migliore possibile.

Per completezza si riporta anche un altra espressione per la funzione d'errore sviluppata in [Niccolini, 2011] e denominata Relative Task Error Function:

$$e_r(q,p,t) = \frac{d^*(q,t)}{D(p,q)+c}$$

con $c \geq 0$ costante per rendere ben definita e_r. Anche con questa espressione restano valide le considerazione svolte precedentemente, tranne per il fatto che il valore ottima di e_r sarebbe 1.

1.3.2 Definizione del problema

Nella Sezione precedente si è introdotto il concetto di TEF e si è concluso che valori minori di questa funzione inducono ad una migliore esecuzione del task.

Risulta perciò naturale considerare il problema di controllo degli agenti come un problema di ottimizzazione con l'obbiettivo di minimizzare $e(q,p,t)$. Tuttavia tale funzione è distribuita nello spazio Q, dati p e t. Per superare questo inconveniente si può integrare e sulla superficie Q e ottenere così un valore scalare che indica quanto bene gli agenti stanno svolgendo il compito a loro assegnato.

È possibile, in forma del tutto generale, definire il seguente funzionale di costo:

$$J(p,t) = \int_Q f\big(e(q,p,t)\big)\sigma(q)\,\mathrm{d}q \tag{1.6}$$

dove $f(\cdot) \geq 0$ è una funzione di costo continua e limitata di $e(q,p,t)$ di cui si parlerà più approfonditamente nella Sezione 1.3.5; invece $\sigma(q) \geq 0$ è una funzione peso dello spazio Q, che può risultare utile per definire aree specifiche di maggior interesse per lo sciame.

La configurazione migliore possibile per lo sciame risulta allora essere:

$$p^*(t) = arg\min_p J(p,t)$$

dove $p = [p_1^T, p_2^T, ..., p_{N_a}^T]^T$ e $p_i = [p_{xi}, p_{yi}, \theta_i]^T$ è il vettore dello stato dell'agente (nel piano).

Risulta perciò necessario trovare una legge di controllo che muova gli agenti verso la configurazione ottima p^*.

Infine, per quanto riguarda le equazioni che caratterizzano gli agenti, il DF Framework così come è stato finora sviluppato, considera una dinamica a singolo integratore ossia:

$$\dot{p}_i = u_i$$

Nel Capitolo 6 si introdurrà una dinamica a doppio integratore.

1.3.3 Controllo a discesa del gradiente

Nella teoria dell'ottimizzazione, una delle metodologie più diffuse per la minimizzazione delle funzioni, prevede di seguire, passo dopo passo, la direzione opposta a quella indicata dal gradiente calcolato in un dato punto.

Senza entrare nel dettaglio di questi algoritmi, è comunque possibile intuire che, se gli agenti si muovessero secondo il gradiente (cambiato di segno) del funzionale J essi arriverebbero a fermarsi in una situazione di minimo (per la precisione si tratta di minimi locali, ma di questo si parlerà in seguito).

Si può perciò scegliere, come legge di controllo per il singolo agente, proprio il gradiente cambiato di segno del funzionale di costo J:

$$\dot{p}_i = u_i = -\beta\frac{\partial J}{\partial p_i} = -\beta\int_Q \frac{\partial f\big(e(p,q)\big)}{\partial p_i}\sigma(q)\,\mathrm{d}q$$

dove β rappresenta il guadagno della legge dei controllo.

In forma più compatta è possibile scrivere tutte le N_a equazioni degli agenti come:

$$\dot{p} = u = -\beta\frac{\partial J}{\partial p} \tag{1.7}$$

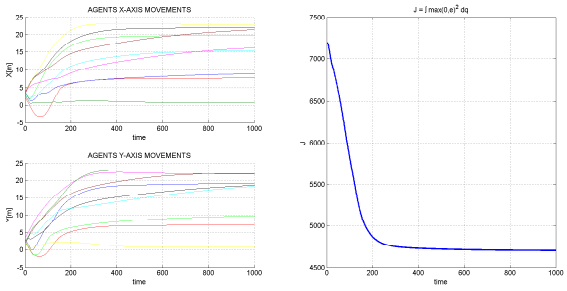


Figura 1.12: Esecuzione del task di Uniform Deployment. **A sinistra**: movimento degli agenti nel tempo lungo gli assi. **A destra**: andamento nel tempo del funzionale di costo J.

.

Si comprende quindi come il moto degli agenti segua quello che risulta essere la direzione di minimo del funzionale J. Tuttavia essendo J non convesso è possibile, e anche molto probabile, che lo sciame si porti in una configurazione stabile di minimo locale. Dal punto di vista puramente analitico si tratta perciò di un controllo sub-ottimo. Per ovviare a questo inconveniente è comunque sempre possibile attuare la serie di metodologie, ormai ampiamente sviluppate ed analizzate nella teoria dell'ottimizzazione, atte ad evitare proprio l'inconveniente dei minimi locali.

In Figura 1.12 è raffigurata l'evoluzione del funzionale di costo J nell'esempio di Uniform Deployment affrontato in Figura 1.10. Si può notare come esso sia non crescente e come sia ininfluente, in termini di buona riuscita del task, che esso non vada a valore nullo.

1.3.4 Analisi di stabilità e convergenza

Per prima cosa è possibile, se non necessario, valutare quali possano essere i punti di stabilità del sistema, ossia quelli per cui si ha uno sciame fermo in una certa configurazione $\bar{p}$. Tali stati sono banalmente quelli per cui si ha:

$$u = \dot{p} = -\beta \frac{\partial J}{\partial p}|_{p=\bar{p}} = 0$$

Questi stati risultano perciò punti di massimo, minimo o sella della funzione J.

È evidente comunque che i punti di sella e di massimo non possano dare origine a configurazioni stabili dello sciame. Infatti, a livello analitico, basterebbe introdurre nel controllo u un disturbo piccolo a piacere per far uscire lo sciame da tale configurazione. A livello pratico anche il semplice errore di quantizzazione, introdotto nel calcolo numerico per il computo di u, è sufficiente ad evitare queste configurazioni indesiderate. Una prova più formale di tali affermazioni si può trovare in [Niccolini, 2011].

Analizzati quindi i punti stazionari $\bar{p}$ è necessario ora dimostrare che essi vengano raggiunti dal sistema.

Per fare ciò [Niccolini, 2011] sviluppa una dimostrazione alla *Lyapunov*, in cui la funzione candidata risulta coincidere proprio con J in quanto semidefinito positivo (infatti sia $f(\cdot)$ che $\sigma(q)$ risultano semidefinite positive). Dimostrando che $\dot{J} \leq 0$ si prova che il funzionale di costo non cresce nel tempo e che risulta nullo solo per i punti $\bar{p}$ definiti in precedenza.

Riconoscendo infine che gli stati $\bar{p}$ costituiscono un insieme invariante, si può ricorrere al *Principio di Invarianza* di LaSalle per dimostrare che le traiettorie del sistema, con la legge di controllo (1.7), convergono ad uno dei punti $\bar{p}$ e che resterà in tale punto in quanto $u|_{p=\bar{p}} = 0$.

Resta quindi da dimostrare che, sotto la legge di controllo (1.7) si ottiene $\dot{J} \leq 0$. Tale prova è leggermente diversa a seconda che il task richiesto sia tempo variante o meno.

Per svolgere le dimostrazioni si può partire da considerazioni preliminari per poi soffermarsi sui particolari relativi ai task. Innanzitutto si calcola $\dot{J}$, ma, per rendere tutto il più chiaro possibile, non si ricorre alle derivate parziali e non si considera il termine $\sigma(q)$ per pura praticità:

$$\begin{aligned}\dot{J} = \frac{\mathrm{d}J}{\mathrm{d}t} &= \int_Q \frac{\mathrm{d}f(e)}{\mathrm{d}t} = \int_Q \frac{\mathrm{d}f(e)}{\mathrm{d}e}\frac{\mathrm{d}e}{\mathrm{d}t} = \int_Q f_e \frac{\mathrm{d}d^* - \sum_i d_i}{\mathrm{d}t} = \\ &= \int_Q f_e \frac{\mathrm{d}d^*}{\mathrm{d}t} - \sum_i \int_Q f_e \frac{\mathrm{d}d_i}{\mathrm{d}t} = \int_Q f_e \frac{\mathrm{d}d^*}{\mathrm{d}t} - \sum_i \left[(\int_Q f_e \frac{\partial d_i}{\partial p_i}) \cdot \dot{p}_i \right] = \\ &= \frac{\partial J}{\partial t} + \frac{\partial J}{\partial p} \cdot \dot{p}\end{aligned}$$

dove $f_e = \frac{\mathrm{d}f(e)}{\mathrm{d}e}$.

Si ha quindi che la parte puramente tempo variante di J dipende da d^*, ossia:

$$\frac{\partial J}{\partial t} = \int_Q f_e \frac{\mathrm{d}d^*}{\mathrm{d}t}.$$

Considerando quindi la legge di controllo (1.7) si ottiene:

$$\dot{J} = \frac{\partial J}{\partial t} - \beta \Big(\frac{\partial J}{\partial p} \Big)^2. \tag{1.8}$$

Per i task tempo invarianti, quali Uniform Deployment, Static coverage e Target Assignment (Sezioni 1.2.2, 1.2.3, 1.2.6 rispettivamente), si ha che $\frac{\partial J}{\partial t} = 0$ e quindi risulta:

$$\dot{J} = -\beta \Big(\frac{\partial J}{\partial p} \Big)^2 \leq 0$$

come volevasi dimostrare.

Per i task tempo varianti, quali l'Effective Coverage e il Dynamic Coverage (Sezioni 1.2.4 e 1.2.5) bisogna valutare nello specifico il termine tempo variante.

Dimostrazione di convergenza per Effective Coverage Task

Si riprende per semplicità l'equazione (1.3) che caratterizza il task:

$$d^*(q,t) = \max\Big(0, \bar{C} - \int_0^t D(p(\tau), q)\,\mathrm{d}\tau\Big) \tag{1.9}$$

Derivando (1.9) nel tempo si ha:

$$\frac{\mathrm{d}d^*}{\mathrm{d}t} = \begin{cases} -\sum_i d_i(p_i(t), q) & \text{quando } \int_0^t \sum_i d_i(p_i(\tau), q)\,\mathrm{d}\tau \geq \bar{C} \\ 0 & \text{altrimenti} \end{cases} \tag{1.10}$$

E quindi:

$$\dot{J} = \underbrace{-\int_Q \overbrace{f_e}^{\geq 0} \cdot \overbrace{\sum_i d_i(p_i(t), q)}^{\geq 0}}_{\geq 0} - \beta\Big(\frac{\partial J}{\partial p}\Big)^2 \leq 0 \tag{1.11}$$

Quindi, e ammettendo che $f_e \geq 0$, senza modificare il controllo u, $\dot{J}$ risulta semidefinita negativa.

Dimostrazione di convergenza per Dynamic Coverage Task

Anche adesso, per motivi di chiarezza, si ripresenta l'equazione (1.4) del task:

$$d^*(q,t) = \max\Big(0, \bar{C} + H(q,t) - \int_0^t D(p(\tau), q)\,\mathrm{d}\tau\Big) \tag{1.12}$$

dove $H(q,t)$ è il termine che fa crescere d^* nel tempo, ad esempio $H = \lambda t$ con $\lambda > 0$; in ogni caso $\dot{H} > 0$. Si deriva quindi l'equazione (1.12):

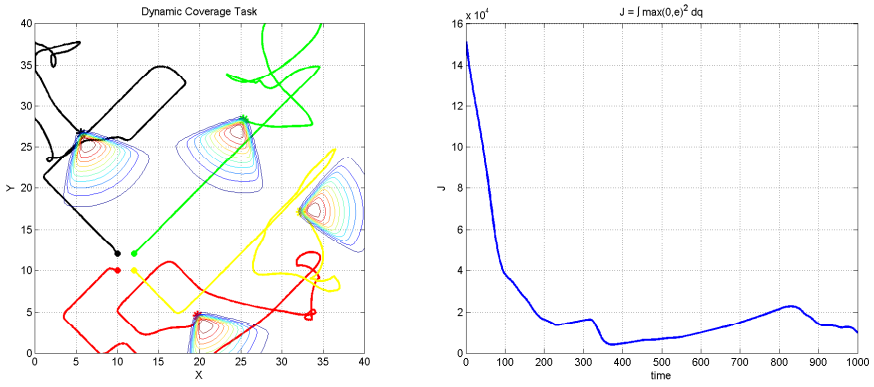
$$\frac{\mathrm{d}d^*}{\mathrm{d}t} = \begin{cases} \dot{H}(q,t) - \sum_i d_i(p_i(t), q) & \text{quando } \int_0^t \sum_i d_i(p_i(\tau), q)\,\mathrm{d}\tau + H(q,t) \geq \bar{C} \\ 0 & \text{altrimenti} \end{cases}$$

E quindi:

$$\dot{J} = \int_Q f_e \cdot (\dot{H} - \sum_i d_i) - \beta\Big(\frac{\partial J}{\partial p}\Big)^2 = \int_Q f_e \dot{H} - \int_Q f_e D(p,q) - \beta\Big(\frac{\partial J}{\partial p}\Big)^2$$

Affinché $\dot{J} \leq 0$ dovrà essere verificato che $\int_Q f_e \dot{H} \leq \int_Q f_e D(p,q)$ cioè d^* non deve crescere troppo velocemente.

Tuttavia è doveroso notare che in questo caso richiedere la convergenza del sistema può non avere senso nel comportamento globale desiderato. Assicurare la convergenza significa garantire che il funzionale di costo non cresca, ma questo non implica che gli agenti non si muovano comunque su tutto l'ambiente ripetutamente, come da programma.



(a) *Percorso compiuto dagli agenti.* **(b)** *Andamento del funzionale di costo J*

Figura 1.13: Dynamic Coverage Task svolto da quattro agenti con FoV gaussiana

Ad esempio, se prendiamo per ipotesi che gli agenti stiano effettivamente pattugliando il territorio, nulla vieta che "cancellino" meno funzione obbiettivo di quanto essa aumenti, il che implica funzionale crescente. Tuttavia questo non modifica il comportamento corretto dello sciame che si muove verso aree in cui la TDF possiede valori più elevati.

In Figura 1.13 si può osservare proprio questo fenomeno: il funzionale di costo non è sempre discendente, tuttavia gli agenti continuano a perlustrare correttamente il territorio.

1.3.5 Scelta della funzione costo

Dalle affermazioni e dalle considerazione svolte nelle Sezioni 1.3.2 e 1.3.3 risulta chiaro come la legge di controllo (1.7) e quindi anche la traiettoria stessa del sistema sia fortemente legata alla forma della funzione di costo $f(\cdot)$.

In generale una delle funzioni di costo più utilizzate in letteratura è quella quadratica:

$$f(e) = e^2. \tag{1.13}$$

Tuttavia, come dimostrato in [Niccolini, 2011], l'utilizzo di tale equazione porta ad avere un elevato numero di minimi locali, effetto alquanto indesiderato.

Un'altra caratteristica indotta da tale funzione di costo è la penalizzazione dell'eccesso di risorse. Se ad esempio lo sciame deve muoversi in una specifica area (Static Coverage Task), con l'equazione (1.13), si avrebbe che gli agenti non si riuniscono tutti assieme dove necessario ma restano distaccati gli uni dagli altri.

Questo comportamento può essere alle volte desiderato ma altre volte controproducente.

Infine un'altra problematica legata alla funzione di costo quadratica e alla penalizzazione dell'eccesso di risorse, riguarda il bordo dell'ambiente Q. Infatti, se la TDF non è sufficientemente elevata ai bordi, allora risulta che gli agenti che si trovano vicino ai confini di Q, tendono ad uscire dalla stessa area, poiché così facendo si ottiene un valore del funzionale J inferiore.

Per sopperire a tutti questi inconvenienti si può utilizzare la seguente forma:

$$f(e) = \max(0, e)^2. \tag{1.14}$$

Adesso la funzione di costo è sempre quadratica, ma non penalizza più l'eccesso di risorse. Infatti laddove la TEF risultasse negativa, la funzione costo resterebbe comunque nulla.

Questa soluzione è la più utilizzata per svolgere le simulazioni e genera, a partire dall'equazione (1.7), la seguente legge di controllo:

$$u_i = 2\beta \int_Q \max(0, e) \frac{\partial d_i(p_i, q)}{\partial p_i} \, \mathrm{d}q$$

Inoltre l'equazione (1.14) garantisce che $f_e = \frac{\mathrm{d}f(e)}{\mathrm{d}e} \geq 0$, il che assicura che $\dot{J} \leq 0$ nell'equazione (1.11) per la dimostrazione di convergenza del task di effective coverage.

Capitolo 2

Obstacle Avoidance

Il Descriptor Functions Framework presentato nel Capitolo 1, sviluppato in [Niccolini, 2011] e ampliato attraverso le Funzioni Descrittive con Field of View in [Ferrari Braga et al., s. d.], risulta una tecnica di controllo di sciami di agenti duttile e applicabile a molti casi pratici differenti.

Una delle caratteristiche sicuramente da menzionare è la semplicità con cui il Framework riesce a gestire task assolutamente differenti: basta modificare, anche durante l'esecuzione stessa del processo, la Task Descriptor Function per ottenere comportamenti dello sciame completamente diversi. È anche possibile unire, modificare o ampliare l'insieme dei task presentati in Sezione 1.2 senza alcuna difficoltà.

Importante ai fini della reale applicabilità del controllo è anche la facilità con cui si riescono a modellare le Funzioni Descrittive degli agenti, così da renderle il più possibile simili a quelle che sono le proprietà/caratteristiche del sensore o del carico di payload montato sugli agenti.

Tuttavia il Framework sviluppato fino a questo punto non offre alcuna specifica riguardo la possibilità di avere, all'interno dell'area di lavoro, degli ostacoli.

Questa mancanza è effettivamente un punto debole. Nei casi pratici sono presenti quasi sempre degli ostacoli che gli agenti dovrebbero accuratamente evitare.

Per rimediare a questo inconveniente Ferrari Braga ha sviluppato, in [Ferrari Braga et al., s. d.], un sistema che, attraverso delle funzioni potenziali, permette agli agenti di muoversi evitando gli ostacoli. Questo metodo genera un controllo u_{obs} che si va a sommare al precedente controllo u_{task} relativo al DF Framework .

u_{obs} viene calcolato come una forza repulsiva applicata sull'agente. È come se l'agente e l'ostacolo fossero due cariche dello stesso segno: tra di essi si viene a formare una forza che spinge l'agente lontano dall'ostacolo.

Questa tecnica, poiché prescinde dal DF Framework, permette di aggiungere, nel calcolo di u_{obs}, anche u_{task} stesso e quindi assicura che l'agente non vada mai a scontrarsi con un ostacolo.

L'uso delle funzioni potenziali per garantire che gli agenti evitino gli ostacoli non è nuovo in letteratura e se ne possono trovare ottimi esempi in [LaValle, 2006] e in [Murphy, 2000]. In [A. Masoud, S. Masoud et al., 2000] si fa un uso molto avanzato delle funzioni potenziali per garantire l'obstacle avoidance e altri vincoli di moto, utilizzando forme analitiche molto particolari e con peculiari proprietà.

Altre tecniche interessanti invece sfruttano quelle che in letteratura sono note come forze giroscopiche. In [Chang et al., 2003] è possibile trovare una applicazione di queste forze ad uno sciame di agenti con dinamica a doppio integratore (non troppo dissimile da quella a singolo integratore del DF Framework).

Le procedure appena descritte, tuttavia, o fanno parte di un framework ben definito oppure si affiancano semplicemente al controllo preesistente, andando così ad accantonare parte delle motivazioni che hanno condotto alla sua sintesi.

Per questa ed altre motivazioni, spiegate in Sezione 2.5, si è scelto di espandere il DF Framework affinché includa, fra le sue capacità, anche l'obstacle avoidance.

2.1 Obstacle Descriptor Functions

All'interno del DF Framework, gli agenti si muovono lungo la direzione indicata dal gradiente del funzionale di costo J, andando così a minimizzare la Task Error Function. Il movimento continua finché lo sciame non si trova in una configurazione di minimo, locale o globale, per cui il controllo, proporzionale al gradiente, risulta nullo.

Lasciando da parte l'aspetto puramente matematico, si può osservare come gli agenti si muovano, singolarmente, verso le aree di Q in cui la funzione obbiettivo d^* ha valori maggiori.

Riportandosi a quanto esposto in precedenza riguardo le funzioni potenziali, è come se la TDF rappresentasse una buca di potenziale verso cui gli agenti sono attratti. L'intensità della forza è proporzionale a d^*.

Ovviamente visto che gli agenti posseggono una ADF limitata nello spazio (nel caso di ADF gaussiana il limite non è matematico ma pratico), l'attrazione si ha solo se la TDF è positiva laddove la Funzione Descrittiva degli agenti è non nulla (per maggior dettagli si veda la Sezione 2.5.1).

Detto ciò, viene spontaneo chiedersi se non sia possibile modellare la TDF così da ottenere delle forze repulsive invece che attrattive. Risolto questo punto l'inserimento di ostacoli all'interno dell'ambiente di lavoro sarebbe il passo immediatamente successivo.

In Sezione 1.3.5, discutendo riguardo le caratteristiche del controllo con l'utilizzo della funzione costo (1.13), si è accennato al fatto che gli agenti tendono ad allontanarsi

quando sono molto vicini fra loro. Questo comportamento indica che nel controllo è presente una componente di repulsione.

In particolare è stato affermato che $f(e) = e^2$ penalizza l'eccesso di risorse, evita cioè che gli agenti stiano vicini tra loro quando la TDF è bassa. Questo accade perché d^* è piccola e, l'intersezione di più ADF, da luogo ad un errore negativo che, passando attraverso la funzione costo, genera un aumento del funzionale J.

Visto che valori positivi della TDF generano forze attrattive, e che la funzione costo (1.13) può generare componenti repulsive, viene naturale chiedersi come si comporterebbero gli agenti in presenza di una TDF negativa.

Si ponga quindi il caso di avere l'agente i-esimo sopra questa ipotetica funzione obbiettivo *negativa*. Si avrebbe $e = d^* - d_i < K_1 < 0$ che attraverso la funzione costo darebbe luogo a $f(e) = K_1^2 > 0$. Viceversa, nel caso in cui invece la ADF non intersecasse d^* si avrebbe: $e = d^* < K_2 < 0$ con $K_1 < K_2 < 0$ e quindi $f(e) = K_2^2 < K_1^2$. Questo implica che il funzionale di costo J sarebbe inferiore nel secondo caso che non nel primo e quindi che l'agente tenderebbe ad allontanarsi dalla zona in cui d^* è negativo.

L'idea, che permette di inserire gli ostacoli all'interno del DF Framework, prevede quindi di descrivere gli ostacoli come funzioni negative dello spazio Q (o meglio, positive ma sottratte nel calcolo dell'errore) attraverso delle Obstacle Descriptor Functions, per brevità ODF.

Accantonando la pura analisi matematica, TDF e ODF si possono interpretare come descrizioni di quanto una zona di Q sia interessante per lo sciame. Così come i valori positivi della TDF indicano un'area molto interessante per gli agenti, così i valori negativi delle ODF rappresentano superfici con scarso valore informativo. Anzi, rappresentano settori in cui muoversi è controproducente, per cui gli agenti tenderanno a non addentrarcisi.

2.1.1 Dettagli matematici

L'aggiunta delle ODF all'interno del Framework comporta la definizione di alcuni aspetti puramente matematici ma assolutamente necessari.

Innanzitutto è necessario individuare la nuova forma della Task Error Function:

$$\begin{aligned} e(q,p,t) &= d^*(q,t) - D(p,q) - d_{obs}(q,p) = \\ &= d^*(q,t) - D(p,q) - \sum_{k=1}^{N_{obs}} d_{obs^k}(q,p) \end{aligned} \tag{2.1}$$

dove $d_{obs^k}(q,p) \geq 0$ rappresenta la ODF del k-esimo ostacolo. Essa è in funzione dello spazio q e, per non perdere di generalità, anche dalla posizione degli agenti p (nelle prossime sezioni si approfondirà nel dettaglio).

La curva rappresentata da d_{obs^k} è, proprio come per le ADF, arbitraria e non è necessario che sia limitata alla sola superficie del k-esimo ostacolo $Q_{obs^k} \subseteq Q_{obs}$. Tuttavia nel seguito si farà sempre riferimento a ODF limitate al solo Q_{obs}, in particolare:

$$d_{obs^k}(q,p) = \begin{cases} h(p) & \text{per } q \in Q_{obs^k} \\ 0 & \text{per } q \notin Q_{obs^k} \end{cases} \tag{2.2}$$

in sostanza, si considera la ODF limitata all'area del ostacolo stesso e con valore $h(p) \geq 0$ uniforme (ma non per forza costante nel tempo).

Detto ciò, risulta necessario modificare anche la TDF. Infatti, se si vuole che $e(q,p,t) \leq 0$ per $q \in Q_{obs}$ e $\forall p, t$ allora la funzione obbiettivo non deve essere presente sugli ostacoli:

$$d^*_{new}(q,p,t) = \begin{cases} d^*_{old}(q,p,t) & \text{per } q \notin Q_{obs^k} \\ 0 & \text{per } q \in Q_{obs^k} \end{cases} \tag{2.3}$$

In questa equazione la nuova forma della Task Descriptor Function è stata indicata con d^*_{new} a sottolineare la differenza con la precedente espressione d^*_{old}, tuttavia nel seguito il pedice *new* verrà scartato.

Per quanto riguarda la funzione di costo, poiché è necessario considerare anche TEF negativa, verrà utilizzata la forma (1.13). Nel seguito tuttavia si introdurrà una nuova equazione per eliminare quelli che sono i difetti della forma quadratica pura, già esposti in Sezione 1.3.5.

2.2 Analisi di stabilità e convergenza

Introdotte le Obstacle Descriptor Functions e fatte le opportune modifiche al Framework risulta necessario verificare quale sia il comportamento del funzionale di costo J, la cui espressione analitica è rimasta immutata (si veda (1.6)).

Si procede perciò allo stesso modo della Sezione 1.3.4, derivando nel tempo J, considerato come candidata di Lyapunov:

$$\dot{J} = \int_Q f_e \frac{\mathrm{d}d^*(q,t)}{\mathrm{d}t} - \int_Q f_e \frac{\mathrm{d}d_{obs}(q,p)}{\mathrm{d}t} - \sum_i \left[(\int_Q f_e \frac{\partial d_i(q,p)}{\partial p_i}) \cdot \dot{p}_i \right]$$

dove ancora $f_e = \frac{\partial f(e)}{\partial e}$. Riguardo a $d_{obs}(q,p)$ si ha:

$$\frac{\mathrm{d}d_{obs}}{dt} = \sum_k \frac{\mathrm{d}d_{obs^k}}{\mathrm{d}t} = \sum_k \frac{\partial d_{obs^k}}{\partial p}\dot{p} = \sum_k \sum_i \frac{\partial d_{obs^k}}{\partial p_i}\dot{p}_i = \sum_i \Big(\sum_k \frac{\partial d_{obs^k}}{\partial p_i} \Big) \dot{p}_i$$

Perciò nel complesso si può riscrivere:

$$\dot{J} = \int_Q f_e \frac{\mathrm{d}d^*(q,t)}{\mathrm{d}t} - \sum_i \left[\Big(\int_Q f_e (\frac{\partial d_i(q,p)}{\partial p_i} + \sum_k \frac{\partial d_{obs^k}(q,p)}{\partial p_i}) \Big) \dot{p}_i \right] = \frac{\partial J}{\partial t} + \frac{\partial J}{\partial p} \cdot \dot{p} \tag{2.4}$$

La precedente legge di controllo (equazione (1.7)), basata sul gradiente del funzionale di costo è ancora valida e conduce al medesimo risultato dell'equazione (1.8):

$$\dot{J} = \frac{\partial J}{\partial t} - \beta\Big(\frac{\partial J}{\partial p}\Big)^2.$$

Esattamente come in 1.3.4 per i task tempo invarianti si ha:

$$\frac{\partial J}{\partial t} = 0 \Rightarrow \dot{J} = -\beta\Big(\frac{\partial J}{\partial p}\Big)^2 \leq 0$$

che indica che il funzionale di costo può solo diminuire fino ad arrivare ad un minimo e quindi si possono ripetere le stesse conclusioni del Capitolo 1 riguardo stabilità e convergenza.

Per quanto riguarda i task tempo varianti è necessario entrare più nel dettaglio, tuttavia la dimostrazione per il Dynamic Coverage Task risulta identica a quella svolta nella Sezione 1.3.4 e non verrà quindi riportata.

Dimostrazione di convergenza per Effective Coverage Task

L'espressione di $\frac{\partial J}{\partial t}$ riguardo al task di effective coverage non è mutata, e l'equazione (1.10), riportata a seguire per chiarezza, è ancora valida:

$$\frac{\mathrm{d}d^*}{\mathrm{d}t} = \begin{cases} -\sum_i d_i(p_i(t), q) & \text{quando } \int_0^t \sum_i d_i(p_i(\tau), q)\,\mathrm{d}\tau \geq \bar{C} \\ 0 & \text{altrimenti} \end{cases}$$

Questa volta però il segno di $f_e = 2e$ non è noto perché l'errore può adesso essere negativo e di conseguenza anche il segno del primo integrale non è noto.
Se però adesso si analizza cosa succede imponendo $u_i = 0 \quad \forall i$, cioè tenendo tutti gli agenti fissi sul posto in cui si trovano, si nota che il funzionale di costo scende.

Infatti richiamando l'equazione (1.3) :

$$d^*(q,t) = \max\Big(0, \bar{C} - \int_0^t D(p(\tau), q)\,\mathrm{d}\tau\Big)$$

si osserva che mentre il tempo scorre, gli agenti "cancellano" la d^* che gli sta attorno, fino a portarla al valore minimo 0. Da quel momento in poi d^* smetterà di decrescere.

Perciò il valore della TDF è sceso in alcuni punti, mentre i valori delle ADF e delle ODF sono rimasti costanti; di conseguenza $e(p,q,t) = d^*(q,t) - D(p,q) - d_{obs}(p,q)$ è sceso in alcuni punti. Quindi il funzionale di costo J è necessariamente diminuito.

Questo fatto implica necessariamente $\dot{J} < 0$ almeno fino a quando d^* non smette di decrescere, ma essendo $u_i = \dot{p_i} = 0$ l'equazione (2.4) risulta composta soltanto dal primo termine:

$$\dot{J}|_{u_i=0} = \frac{\partial J}{\partial t} = -\int_Q f_e \cdot \sum_i d_i(p_i(t), q)\,\mathrm{d}q$$

che risulta quindi essere negativa.

Decretato perciò il segno negativo del termine tempo variante di J si può affermare che $\dot{J} \leq 0$.

2.2.1 Analisi della legge di controllo

Nelle precedenti pagine è stato dimostrato come, anche dopo aver aggiunto le ODF, una legge di controllo a discesa del gradiente garantisca che lo sciame esegua il compito affidatogli.

Tuttavia osservando l'equazione del gradiente di J per esteso, ci si accorge di un termine aggiuntivo rispetto al caso del Capitolo 1:

$$\begin{aligned} u_i = \dot{p}_i = -\beta \frac{\partial J}{\partial p_i} &= \beta \int_Q f_e \frac{\partial (d_i(p,q) + \sum_k d_{obs^k}(p,q))}{\partial p_i} \, \mathrm{d}q = \\ &= \beta \int_Q f_e \frac{\partial d_i(q,p)}{\partial p_i} \, \mathrm{d}q + \beta \int_Q f_e \sum_k \frac{\partial d_{obs^k}(q,p)}{\partial p_i} \, \mathrm{d}q \end{aligned} \tag{2.5}$$

Il nuovo elemento aggiunto alla legge di controllo (2.5) non è il diretto responsabile dell'obstacle avoidance. L'effetto degli ostacoli si riflette principalmente in f_e che diventa sempre più negativa man mano che l'agente si avvicina all'ostacolo, col risultato di respingere l'agente. Il fattore aggiunto ad u_i assicura $\dot{J}$ negativo e aiuta gli agenti ad evitare gli ostacoli prima che essi si avvicinino troppo.

In Figura 2.1 è mostrato proprio questo effetto applicato ad uno Static Coverage Task con alta dispersione lungo x. Le Figure 2.1a e 2.1b mostrano rispettivamente la traiettoria degli agenti e l'andamento di $\dot{J}$ nel caso cui non fosse presente il secondo termine nel controllo (2.5).

Si nota che la derivata del funzionale di costo non è sempre negativa. In questo caso ciò non costituisce un problema in quanto gli agenti riescono comunque a concludere il task, ma in generale questo potrebbe non essere sempre vero.

Raffrontando inoltre il percorso effettuato con quello presente in Figura 2.1c, in cui la legge di controllo è completa, si nota come quest'ultimo sia ben distante dagli ostacoli, garantendo così margini di sicurezza maggiori.

Infine questo termine dà percezione dell'ostacolo anche quando esso non si trova all'interno della DF dell'agente, perché prescinde dalla DF stessa.

2.3 Obstacle avoidance e modellazione matematica degli ostacoli

Dopo aver introdotto le Obstacle Descriptor Functions, aver discusso su come esse agiscano e per quali motivi gli agenti vengano respinti da esse, si è dimostrato che il controllo a discesa del gradiente permette ancora di avere convergenza dell'algoritmo

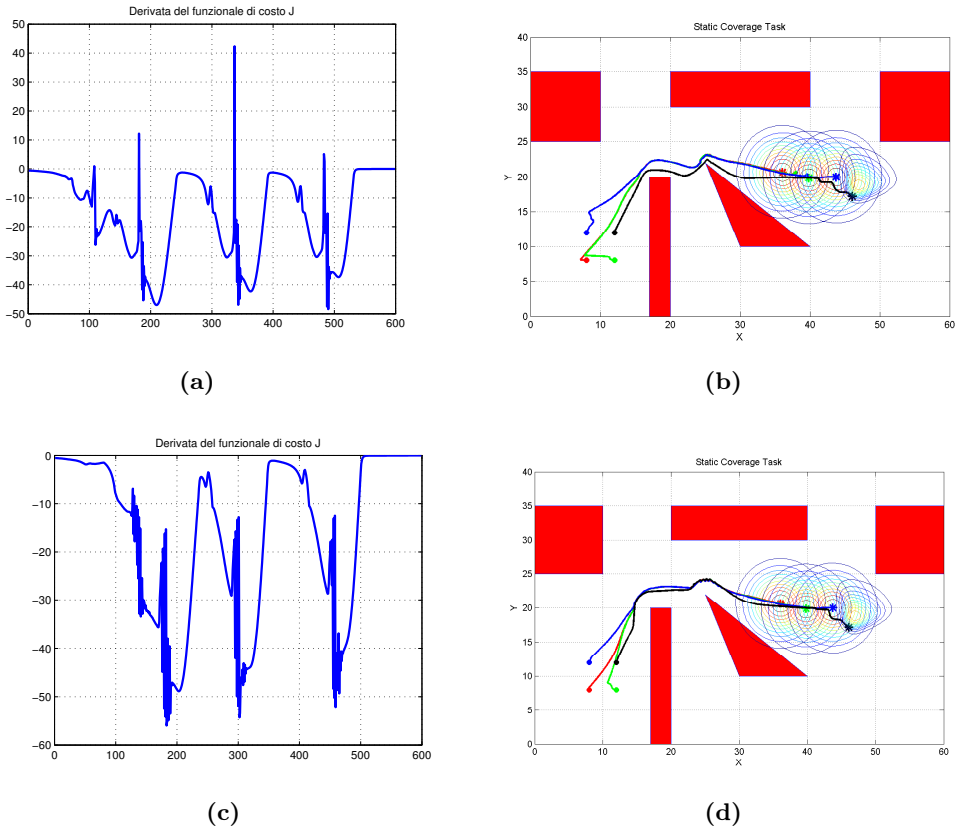


Figura 2.1: Esempio di Static Coverage Task (con alta dispersione lungo x). Le Figure 2.1a e 2.1b considerano il caso in cui all'equazione (2.5) manchi il secondo termine. Le restanti Figure 2.1c e 2.1d invece implementano la legge di controllo completa.

verso un minimo della TEF, portando quindi a compimento il task assegnato allo sciame.

Tuttavia non è stata fornita alcuna dimostrazione che certifichi che gli agenti non collidano contro gli ostacoli. È certo che l'agente sarà soggetto ad una specie di forza repulsiva, ma nulla vieta che tale contributo sia inferiore alla componente attrattiva che muove l'agente verso il task.

Potrebbero perciò presentarsi situazioni in cui gli agenti, fortemente attirati dalla funzione obbiettivo, finiscano per scontrarsi con gli ostacoli.

Un soluzione potrebbe essere quella di aumentare le ODF così da ottenere sicuramente una componente repulsiva maggiore e quindi evitare le collisioni. Una strategia di questo tipo conduce purtroppo ad un comportamento indesiderato: gli agenti piuttosto che muoversi in direzione del task, si spostano lontano dagli ostacoli.

Si può allora pensare ad una funzione che cresca in ampiezza man mano che gli agenti si avvicinano ad essa. Meglio ancora: l'ampiezza potrebbe tendere all'infinito quando l'agente tocca l'ostacolo.

Proposizione 1. *Se $d_{obs^k} \to \infty$ quando la posizione di almeno un agente tende al bordo dell'ostacolo k-esimo, allora gli agenti non collideranno mai contro tale ostacolo.*

Dimostrazione. Se $d_{obs^k} \to +\infty$ allora:

$$e(p, q, t) = d^*(q, t) - D(p, q) - \sum_k d_{obs^k}(q, p) \to -\infty$$

e quindi

$$J = \int_Q f(e)\,\mathrm{d}q \to +\infty.$$

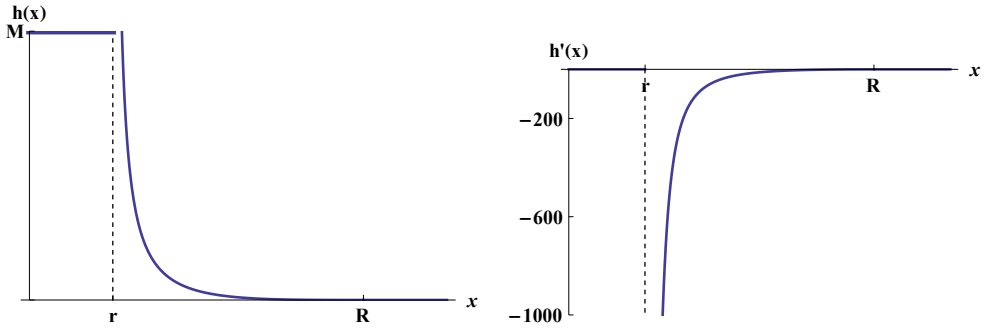
Tuttavia è stato precedentemente dimostrato che $\dot{J} \leq 0$. Ciò implica che J non può crescere all'infinito, può solo decrescere o restare costante. Quindi non è possibile che $d_{obs^k} \to +\infty$ e questo significa che gli agenti non possono toccare il bordo degli ostacoli. □

Resta quindi da definire come si possa variare la Obstacle Descriptor Function in relazione alla posizione degli agenti.

Una valida scelta può essere la seguente:

$$d_{obs^k} = \begin{cases} \sum_{i=1}^{N_a} h(\|p_i - {}^k c_i\|) & \text{per } q \in Q_{obs^k} \\ 0 & \text{per } q \notin Q_{obs^k} \end{cases} \tag{2.6}$$

dove ${}^k c_i$ è il punto sul bordo dell'ostacolo k-esimo più vicino all'agente i-esimo. Inoltre, vista la precedente proposizione 1, dovrà essere che $h(\|p_i - {}^k c_i\|) \to +\infty$ per $\|p_i - {}^k c_i\| \to 0^+$ ossia quando l'agente si trova sul bordo dell'ostacolo.



(a) $h(x)$ **(b)** $h'(x)$

Figura 2.2: Grafico della funzione $h(\cdot)$ e della sua derivata con $r = 2$ e $R = 8$.

Come si può notare, è stata scelta una d_{obs^k} che ha valori non nulli solo su Q_{obs^k} e su questo insieme è costante (cioè ha lo stesso valore $\forall q \in Q_{obs^k}$).

La ODF del k-esimo ostacolo dipende dalla posizione di tutti gli agenti e di conseguenza è tempo variante.

Resta ora da decidere la forma della funzione $h(\cdot)$ in modo che soddisfi le caratteristiche richieste. Un esempio può essere:

$$h(x) = \begin{cases} \max(0, -\frac{(x-R)^3}{(x-r)}) & \text{per } x > r \\ M & \text{per } x \leq r \end{cases} \tag{2.7}$$

con $0 \leq r < R < +\infty$ e M un valore positivo grande a piacere.

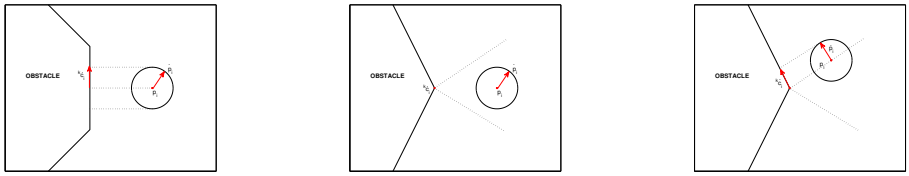
Come si vede in Figura 2.2a, $h(\cdot)$ è una funzione continua e derivabile, diversa da 0 solo nell'intervallo $]0, R]$ e con asintoto verticale in r.

La particolare forma di $h(\cdot)$ permette di scegliere, regolando r, quanto gli agenti debbano stare lontani dall'ostacolo: la distanza fra agente e bordo dell'ostacolo sarà sicuramente maggiore di r.

Inoltre se l'agente i-esimo dista dall'ostacolo più di R, allora quell'agente non influenza d_{obs^k} perché $h(x) = 0$ per $x \geq R$. In questo modo abbiamo che gli ostacoli distanti dagli agenti hanno valore nullo e non interferiscono minimamente con il moto dello sciame.

2.3.1 Caratteristiche della nuova legge di controllo

Prima di analizzare nel dettaglio il funzionamento del controllore ottenuto aggiungendo le ODF, è necessario valutare l'effettiva forma di $\dot{J}$ (equazione (2.4)), adesso che d_{obs^k} è stato completamente caratterizzato. In particolare deve esserne valutata la derivata temporale, che dà in seguito luogo alla nuova componente del controllo



(a) **(b)** **(c)**

Figura 2.3: Possibili casi in cui può trovarsi un agente rispettivamente ad un ostacolo.

u_i^{obs}.

$$\frac{\mathrm{d}d_{obs^k}}{\mathrm{d}t} = \begin{cases} \sum_i h'(\|p_i - {}^k c_i\|) \dfrac{p_i - {}^k c_i}{\|p_i - {}^k c_i\|} \cdot (\dot{p_i} - {}^k\dot{c_i}) & q \in Q_{obs^k} \\ 0 & q \notin Q_{obs^k} \end{cases} \tag{2.8}$$

Nella derivata appena calcolata appare, come previsto, il controllo $u_i = \dot{p_i}$. Tuttavia è presente anche un termine che, nella precedente dimostrazione sulla stabilità del sistema, non era stato considerato, ossia ${}^k\dot{c_i}$.

${}^k c_i$ rappresenta, come già detto, il punto, sul bordo dell'ostacolo k-esimo, più vicino a p_i. La sua derivata temporale è perciò da intendersi in funzione di p_i, infatti se l'agente si sposta allora ${}^k c_i$ potrebbe variare:

$${}^k\dot{c_i} = \frac{\partial {}^k c_i}{\partial p_i} \cdot \dot{p_i}$$

Per capire meglio come sia fatto ${}^k\dot{c_i}$ si pensi a cosa succede se l'agente si sposta in un intorno della sua posizione p_i. La Figura 2.3 mostra i tre possibili casi:

Figura 2.3a : il punto ${}^k c_i$ è sul lato dell'ostacolo. Se l'agente si sposta perpendicolarmente al lato dell'ostacolo allora ${}^k\dot{c_i} = 0$, altrimenti risulta essere un vettore parallelo al lato stesso.

Figura 2.3b : il punto ${}^k c_i$ è sul vertice dell'ostacolo. In questo caso, ${}^k\dot{c_i} = 0$ finché l'agente si muove all'interno del cono duale (indicato con linee tratteggiate) generato dai due lati che formano il vertice.

Figura 2.3c : uguale al caso precedente ma questa volta l'agente si sposta fuori dal cono duale (indicato con linee tratteggiate) puntato sul vertice. Anche in questo caso ${}^k\dot{c_i}$ risulta essere parallelo al lato dell'ostacolo.

In tutti i casi descritti, ${}^k\dot{c_i}$ è nullo oppure è parallelo al lato dell'ostacolo. Espandendo l'equazione (2.8) si ottiene:

$$\frac{\mathrm{d}d_{obs^k}}{\mathrm{d}t} = \sum_i \frac{h'(\|p_i - {}^k c_i\|)}{\|p_i - {}^k c_i\|} \Big[(p_i - {}^k c_i) \cdot \dot{p_i} - (p_i - {}^k c_i) \cdot {}^k\dot{c_i} \Big]$$

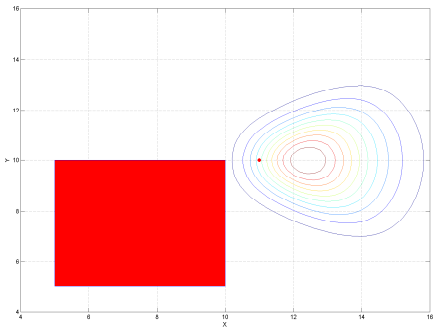


Figura 2.4: La DF dell'agente è praticamente nulla sull'ostacolo. Perciò l'effetto di quest'ultimo su u_i^{task} è a sua volta nullo.

L'espressione $p_i - {}^k c_i$ rappresenta un vettore che parte da ${}^k c_i$ e arriva in p_i, e quindi, per costruzione, sempre perpendicolare al lato dell'ostacolo.

Perciò nei casi in cui ${}^k\dot{c}_i \neq 0$ risulta comunque che $(p_i - {}^k c_i) \cdot {}^k\dot{c}_i = 0$ in quanto prodotto scalare di vettori perpendicolari e quindi l'equazione (2.8) si può riscrivere come:

$$\frac{\mathrm{d}d_{obs^k}}{\mathrm{d}t} = \sum_i \frac{h'(\|p_i - {}^k c_i\|)}{\|p_i - {}^k c_i\|}(p_i - {}^k c_i) \cdot \dot{p}_i. \tag{2.9}$$

eliminando di fatto ${}^k\dot{c}_i$, termine effettivamente scomodo da calcolare.

Si può adesso analizzare il nuovo controllore (equazione (2.5)), ottenuto dall'aggiunta delle ODF nel framework:

$$\begin{aligned} \dot{p}_i = u_i &= \beta \int_Q f_e \frac{\partial d_i}{\partial p_i} + \beta \sum_k \int_Q f_e \frac{\partial d_{obs^k}}{\partial p_i} = \\ &= \beta \int_Q f_e \frac{\partial d_i}{\partial p_i} + \beta \sum_k \int_{Q_{obs^k}} f_e \frac{h'(\|p_i - {}^k c_i\|)}{\|p_i - {}^k c_i\|}(p_i - {}^k c_i) = \\ &= \beta[u_i^{task} + u_i^{obs}] \end{aligned} \tag{2.10}$$

nella seconda uguaglianza, il secondo integrale è su Q_{obs^k} in quanto solo su questo spazio $\frac{\partial d_{obs^k}}{\partial p_i} \neq 0$.

È importante notare come l'effetto di d_{obs^k} non sia presente solo nel termine u_i^{obs} del controllo, ma bensì anche in $f_e = e^2$ e quindi anche in u_i^{task}.

L'effetto dell'ostacolo sul termine u_i^{task} dipende molto dalla DF dell'agente. Infatti se $d_i(q, p_i) \gg 0$ per $q \in Q_{obs^k}$ allora l'ostacolo genera una azione repulsiva sull'agente.

Tuttavia l'azione dell'ostacolo sull'agente attraverso u_i^{task} è sostanzialmente nulla nel caso in cui $d_i(q, p_i) \simeq 0$ per $q \in Q_{obs^k}$. Una situazione di questo tipo è rappresentata in Figura 2.4.

Si può in definitiva affermare che gli agenti distanti dal k-esimo ostacolo, non solo non ne modificano la ODF, ma non ne risentono nemmeno gli effetti in u_i^{task}. Tuttavia

non è vero il viceversa: è possibile che u_i^{task} non venga influenzato dall'ostacolo anche se l'agente è vicino ad esso (Figura 2.4).

Si analizza adesso il termine :

$$u_i^{obs} = \sum_k \int_{Q_{obs^k}} f_e \frac{h'(\|p_i - {}^k c_i\|)}{\|p_i - {}^k c_i\|}(p_i - {}^k c_i) \tag{2.11}$$

Si può subito notare come u_i^{obs} sia totalmente indipendente dalla DF dell'agente. Dipende piuttosto dalla distanza fra tutti gli ostacoli e l'agente in questione.

In particolare se $h'(\|p_i - {}^k c_i\|) = 0, \forall k = 1, ..., N_{obs}$ allora $u_i^{obs} = 0$.

Questa condizione si verifica per $\|p_i - {}^k c_i\| \geq R, \forall k = 1, ..., N_{obs}$, ossia quando l'agente è sufficientemente distante da tutti gli ostacoli e quindi, a buon ragione, non necessita di una componente di controllo per l'obstacle avoidance.

Si consideri adesso, per motivi di semplicità, che ci sia solo l'ostacolo $\bar{k}$ vicino all'agente (o che tutti gli altri ostacoli siano abbastanza lontani).

$$u_i^{obs} = \frac{h'(\|p_i - {}^{\bar{k}} c_i\|)}{\|p_i - {}^{\bar{k}} c_i\|}(p_i - {}^{\bar{k}} c_i) \int_{Q_{obs^{\bar{k}}}} f_e \, \mathrm{d}q$$

poiché l'integrale è su $Q_{obs^{\bar{k}}}$, $\quad f_e = -d_i - d_{obs^{\bar{k}}}$ in quanto $d^* = 0$ per $q \notin Q_{free}$ e allora:

$$\begin{aligned} \int_{Q_{obs^{\bar{k}}}} f_e \, \mathrm{d}q &= -\int_{Q_{obs^{\bar{k}}}} d_i(p,q) \, \mathrm{d}q - \int_{Q_{obs^{\bar{k}}}} h(\|p_i - {}^{\bar{k}} c_i\|) \, \mathrm{d}q = \\ &= -\int_{Q_{obs^{\bar{k}}}} d_i(p,q) \, \mathrm{d}q - h(\|p_i - {}^{\bar{k}} c_i\|) \int_{Q_{obs^{\bar{k}}}} \mathrm{d}q = \\ &= -h(\|p_i - {}^{\bar{k}} c_i\|) A_{obs^{\bar{k}}} - \int_{Q_{obs^{\bar{k}}}} d_i(p,q) \, \mathrm{d}q \end{aligned} \tag{2.12}$$

dove $A_{obs^{\bar{k}}}$ è l'area coperta dall'ostacolo. Si può infine riscrivere:

$$u_i^{obs} = -\frac{h'(\|p_i - {}^{\bar{k}} c_i\|)}{\|p_i - {}^{\bar{k}} c_i\|} \left(h(\|p_i - {}^{\bar{k}} c_i\|) A_{obs^{\bar{k}}} + \int_{Q_{obs^{\bar{k}}}} d_i \right) (p_i - {}^{\bar{k}} c_i) \tag{2.13}$$

e ricordando che $h(\cdot) \geq 0$ (si veda Figura 2.2a), $h'(\cdot) \leq 0$ (si veda Figura 2.2b) e $d_i \geq 0$ si ottiene:

$$u_i^{obs} = \gamma \frac{p_i - {}^{\bar{k}} c_i}{\|p_i - {}^{\bar{k}} c_i\|}, \qquad \text{con } \gamma \geq 0 \tag{2.14}$$

ossia una componente di controllo di modulo γ e con direzione e verso uguale a quella di $(p_i - {}^{\bar{k}} c_i)$, che non è altro che un vettore perpendicolare alla superficie dell'ostacolo (o puntato su un suo vertice) uscente da esso.

Come era giusto aspettarsi, u_i^{obs} è una componente che cerca sempre di allontanare l'agente dagli ostacoli (a distanza minore di R) che ha attorno.

In conclusione u_i si compone di due termini, u_i^{task} e u_i^{obs}. Sul primo l'obstacle DF ha effetto solo se la funzione descrittiva dell'agente si trova almeno in parte sull'ostacolo. Il secondo invece dà sempre origine ad un'azione volta ad evitare l'ostacolo quando esso è sufficientemente vicino all'agente, a prescindere dalla DF di quest'ultimo.

2.4 Funzione di costo alternativa

Durante tutto lo svolgimento dell'analisi di convergenza e della legge di controllo che si ottiene dall'introduzione delle ODF è sempre stata utilizzata la funzione di costo quadratica presentata in Sezione 1.3.5. Il motivo principale di questa scelta, già spiegato in precedenza, è dovuto al fatto che è necessario che i valori negativi della TEF vengano considerata nel calcolo del funzionale J.

La funzione costo dell'equazione (1.14) eliminando le componenti negative dell'errore cancellerebbe la componente repulsiva presente all'interno di u^{task} e anche u^{obs} risulterebbe nullo, rendendo di fatto impossibile evitare le collisioni.

Tuttavia, come già sottolineato in Sezione 1.3.5, la forma quadratica pura $f(e) = e^2$, comporta un sacco di inconvenienti. Oltre al fatto di penalizzare l'eccesso di risorse, di dar luogo ad un elevato numero di minimi locali, comporta anche che gli agenti vicini al bordo di Q tendano ad uscire dall'ambiente di lavoro se la TDF non ha valori sufficientemente elevati.

Le soluzioni a tale inconveniente sono molteplici: si può aumentare la TDF, considerare Q più ampio rispetto all'effettiva area di lavoro, oppure si possono collocare, lungo tutto il confine, degli ostacoli virtuali, così da allontanare gli agenti.

Tutte le soluzioni proposte, tuttavia, comportano degli inconvenienti, come task sovradimensionati, maggior costo computazionale, l'impossibilità di porre task su tutto l'ambiente Q.

La funzione costo dell'equazione (1.14) superava analiticamente le problematiche sopracitate ma, eliminando i valori negativi dell'errore, andrebbe a cancellerebbe la componente repulsiva presente all'interno di u^{task}.

Generalmente parlando $e(q, p, t)$ può assumere qualsiasi valore in qualunque punto $q \in Q$. Tuttavia ai fini dell'obstacle avoidance è sufficiente che l'errore sia negativo laddove è presente l'ostacolo. Perciò la vera condizione da rispettare è che $e(q, p, t) \leq 0$ per $q \in Q_{obs}$.

Si può perciò modificare l'espressione (1.14) per fare in modo che tenga in considerazione i valori negativi solo per $q \in Q_{obs}$.

Per fare ciò si considera la seguente funzione costante nel tempo:

$$g(q) = \begin{cases} 0 & \text{per } q \notin Q_{obs} \\ -M & \text{per } q \in Q_{obs} \end{cases} \tag{2.15}$$

dove $M > 0$ è un valore grande a piacere, anche infinito.

Adesso si può definire la nuova funzione costo:

$$f(e(q,p,t)) = \max(g(q), e(q,p,t))^2 \tag{2.16}$$

In questo modo laddove non sono presenti ostacoli, i valori della TEF negativi, vengono riportati a zero; invece su Q_{obs} l'errore può essere negativo al più fino al valore $-M$. Scegliendo M infinito non si hanno limitazioni su quanto la TEF possa essere negativa sugli ostacoli.

Questa nuova funzione di costo mantiene tutte le proprietà di quella puramente quadratica, ma ne elimina gli svantaggi, al costo di una maggior complessità analitica.

Si vuole sottolineare inoltre che tutte le dimostrazioni svolte, che utilizzavano $f(e) = e^2$, possono essere ripetute con la nuova forma (2.16). Tale espressione infatti non comporta alcun problema analitico, neanche nei delicati passaggi di derivazione. Infatti essendo (2.15) indipendente dalla pozione degli agenti, essa non entra mai in gioco nei processi di derivazione.

In tutti gli esempi svoli durante il corso del capitolo si è sempre utilizzata questa nuova funzione di costo.

2.5 Conclusioni ed esempi

In definitiva all'interno del capitolo è stato modificato parte del DF Framework per includere quella che in letteratura è nota come *obstacle avoidance*.

Il lavoro svolto, rispetto alla banale aggiunta di un secondo controllo che garantisca che gli agenti non collidano con gli ostacoli, non altera le caratteristiche di sub-ottimalità proprie del DF Framework.

Infatti, il Framework, grazie alla realizzazione di un funzionale di costo intrinsecamente legato al payload degli agenti, genera percorsi che sono ottimi dal punto di vista dell'utilizzo del payload stesso. Per uno stesso task, con le stesse condizioni iniziali, si ottengono traiettorie diverse a seconda che gli agenti posseggano ADF omnidirezionale o con FoV.

È proprio da questo punto di vista che l'aggiunta dell'obstacle avoidance all'interno del Framework è un passo importante. L'agente, nelle soluzioni precedenti (si veda [Ferrari Braga, 2013]), vedeva l'ostacolo solo come un oggetto da scansare. Invece adesso lo sciame evita gli ostacoli per non sprecare le potenzialità del payload su un target di scarso interesse.

Si consideri ad esempio un agente con una telecamera. Con il controllo attuale, in prossimità di un ostacolo, l'agente ruota il suo cono di vista fuori dall'ostacolo, puntandolo anzi sull'ambiente di lavoro Q_{free} da cui deve ricavare informazioni.

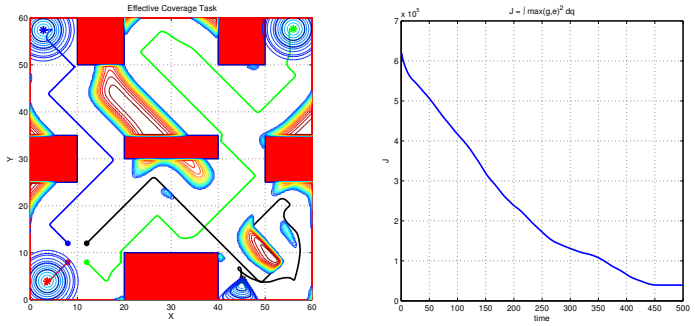


Figura 2.5: Effective Coverage Task. Le curve di livello mostrano la TDF alla fine del task.

2.5.1 Commenti sul task Effective Coverage

Il task di Effective Coverage è il più particolare fra quelli mostrati finora. Infatti, solo in questo caso, la funzione obbiettivo decresce senza poter mai risalire.

Questo comportamento implica innanzitutto che il task sarebbe completato per intero solo quando la TDF risultasse nulla su tutto Q. Utilizzando inoltre la funzione costo (2.16), si avrebbe che il minimo globale del funzionale J varrebbe proprio 0. Tuttavia la realtà dei fatti mostra che lo sciame si arresta ben prima di raggiungere il risultato ottimo.

Ciò è dovuto alla limitatezza delle ADF. Escludendo il termine u^{obs}, si analizza la restante parte della legge di controllo:

$$u_i^{task} = \beta \int_Q f_e \frac{\partial d_i(p,q)}{\partial p_i} \, \mathrm{d}q \tag{2.17}$$

Il temine $d_i(p,q)$ è limitato nello spazio, perciò anche la sua derivata parziale rispetto a p_i risulta tale. Ciò significa che, della funzione $f(e)$, gli unici valori considerati sono quelli in cui la ADF non è nulla. È come se l'integrale non fosse su tutto Q ma solo per le $\bar{q}$ tali che $d_i(p,\bar{q}) \neq 0$.

Questo aspetto, estremamente importante, è vero sempre, a prescindere dal task e dalla presenza di ostacoli, ed è già stato menzionato in Sezione 2.3.1 e nella Figura 2.4.

È proprio questa caratteristica che limita le prestazioni dello sciame nell'effective coverage. Si osservi ad esempio la Figura 2.5 e in particolare l'agente rosso. Esso si muove, rispetto agli altri agenti, molto poco. Il motivo è che tutto attorno alla sua posizione la TEF è nulla in quanto "cancellata" dagli agenti ad inizio missione. Perciò il controllo, che muove l'agente, risulta nullo.

La stessa cosa accade anche agli altri agenti: rendendo nulla la TEF attorno a sé, gli agenti si fermano. Per questo motivo restano zone di Q in cui la TDF non è nulla.

Questo problema è stato riscontrato anche da Hussein in [Hussein, Stipanović et al., 2007a]. Anche nel suo studio gli agenti soffrono del medesimo problema. La soluzione proposta prevede di applicare un controllo secondario che, quando il primo risulta nullo, muove l'agente verso zone in cui la funzione obbiettivo sia già stata cancellata.

Nel Capitolo 3 si vedrà come, imponendo agli agenti di restare compatti tra di loro, si possono ottenere risultati migliori di quelli attuali.

2.5.2 Deadlock

Purtroppo il Framework sviluppato finora soffre del problema del deadlock: può capitare che gli agenti si fermino davanti ad uno ostacolo senza riuscire a superarlo.

Tale inconveniente si presenta quando l'obbiettivo dell'agente è proprio dietro ad un ostacolo. Un esempio di questo fenomeno è riportato in Figura 2.6a. L'agente deve svolgere un task di static coverage posizionato oltre l'ostacolo. Purtroppo il problema non è dovuto solo alla perfetta simmetria dell'esempio: la posizione finale in cui si ferma l'agente è proprio un minimo locale. Infatti anche nell'esempio di Figura 2.6b l'agente, pur non partendo da una condizione di simmetria come nel caso precedente, finisce per arrivare nella stessa posizione.

Inoltre non è semplice neppure individuare i casi in cui questo fenomeno avviene. In Figura 2.6c gli agenti superano correttamente l'ostacolo, mentre in Figura 2.6d uno dei tre resta in situazione di deadlock.

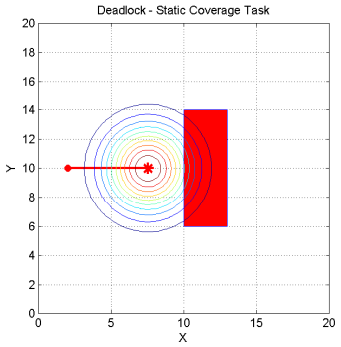
Risulta quindi davvero difficile individuare come modificare il DF Framework così da evitare situazioni di deadlock. In questi casi particolari conviene rinunciare ai criteri di ottimalità e sfruttare controllori secondari, in parallelo ai primi, che aiutino a superare la situazione di stallo.

2.5.3 Obstacle detection

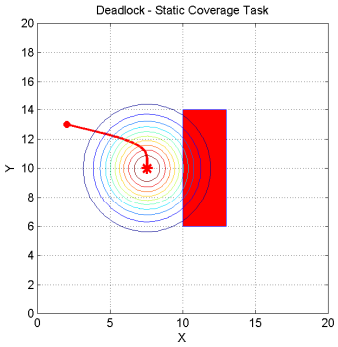
La legge di controllo sviluppata per garantire l'obstacle avoidance presuppone di lavorare in un ambiente strutturato, ossia in cui è nota la locazione esatta degli ostacoli.

Tuttavia è possibile anche fare in modo che gli agenti rilevino gli ostacoli nell'ambiente senza conoscerne prima la posizione. Ciò è possibile grazie alla flessibilità fornita dalla legge che modifica le ODF.

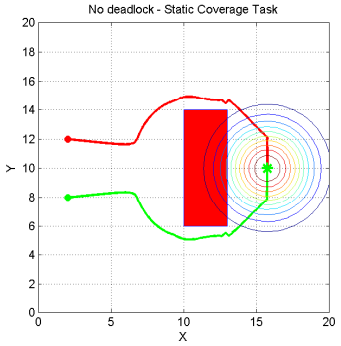
Infatti ammesso che gli agenti riescano ad identificare la presenza di un ostacolo ad una distanza superiore a R, allora $d_{obs} = 0$ (dall'equazione (2.6)) e quindi il loro effetto sull'agente sarebbe stato comunque nullo anche se fosse stato un ostacolo noto.



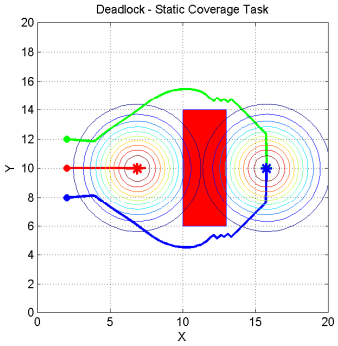
(a) *Situazione di deadlock*



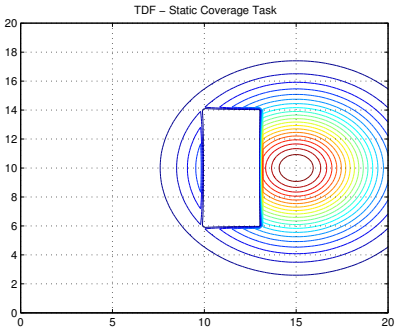
(b) *Situazione di deadlock*



(c) *Nessun deadlock*

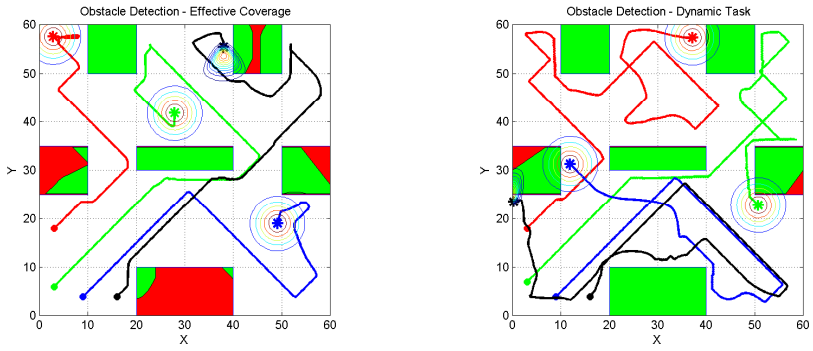


(d) *Situazione di deadlock parziale*



(e) *TDF dello Static Coverage Task*

Figura 2.6: Varie situazioni di deadlock e non, applicate ad una caso di Static Coverage



(a) *Effective Coverage* **(b)** *Dynamic Coverage*

Figura 2.7: Applicazione di obstacle detection utilizzando due task differenti. Gli ostacoli ignoti agli agenti sono rappresentati in rosso. In verde invece sono raffigurate le porzioni individuate dallo sciame.

In pratica l'ostacolo viene riconosciuto prima che esso possa avere alcun effetto sull'agente, questo garantisce un comportamento soft. Se così non fosse si avrebbe che, al momento in cui l'ostacolo viene scoperto, u^{obs} sarebbe non nullo e lo stesso per la componente repulsiva in u^{task}. Si avrebbe cioè un picco nel controllo dell'agente che potrebbe anche portare all'instabilità.

Per attuare quindi il task di obstacle detection, basta eseguirne uno di effective coverage e considerare un ostacolo come individuato quando entra nel campo di vista dell'agente superando una certa soglia della ADF. Ovviamente soltanto una porzione dell'ostacolo viene identificata, ma quando più porzioni si intersecano esse vengono unite in un unico ostacolo.

In Figura 2.7 è possibile osservare il percorso effettuato dagli agenti. Le porzioni di ostacoli individuati sono rappresentate in verde, invece quelle in rosso rappresentano l'ostacolo reale, non visto dallo sciame.

Si può notare come alcuni ostacoli vengano identificati come maggior accuratezza di altri, ad esempio quello centrale e quelli in alto. Gli altri ostacoli sono per metà, se non di più, ignoti allo sciame. Questo è dovuto al fatto che il task di effective coverage è stato per quegli ostacoli poco efficiente. Muovendo gli agenti con un dynamic coverage, Figura 2.7b, si è sicuri di perlustrare correttamente tutta l'area.

Una particolarità di questo task è quella di aver considerato le Funzioni Descrittive degli agenti non più come una descrizione astratta del sensore di payload, ma come una rappresentazione fisica. In questo caso l'informazione fornita dal sensore viene effettivamente sfruttata per agire, in un secondo momento, sul controllo. In tutti gli altri task, l'informazione acquisita dal payload, non influenza il controllo, ma va a vantaggio di una terza parte che la sfrutterà in altro modo.

Capitolo 3

Vincoli sulle distanze inter-agente

Il Descriptor Functions Framework analizzato finora risulta essere ben più di una tecnica di controllo. Esso, descrivendo analiticamente la caratteristica del payload degli agenti, permette di avere un movimento complessivo dello sciame che va ad ottimizzare le capacità dell'intera squadra di agenti. Inoltre non si focalizza su un singolo obbiettivo: i task da assegnare allo sciame sono molteplici e per cambiare dall'uno all'altro è sufficiente modificare un singolo parametro.

In aggiunta a ciò, avendo incluso all'interno del Framework stesso la presenza degli ostacoli, modellati come Funzioni Descrittive che si sottraggono nel calcolo dell'errore, si è riusciti non solo a garantire che gli agenti evitino gli ostacoli, ma anche che non sprechino risorse del proprio payload su un obbiettivo di nessun interesse.

Tuttavia se gli agenti sono ora in grado di evitare gli ostacoli, nulla vieta che non possano scontrarsi tra loro stessi.

Per superare anche questo problema si potrebbe pensare di ripetere il ragionamento seguito per costruire l'obstacle avoidance. Purtroppo in questo caso gli agenti si dovrebbero vedere fra di loro come ostacoli, ossia la TEF dovrebbe essere negativa proprio sugli agenti stessi, per poter aver un contributo repulsivo all'interno della legge di controllo.

Un simile effetto lo si può ottenere utilizzando la funzione $f(e) = e^2$. Tuttavia non si assicura comunque che gli agenti si evitino fra di loro.

Oltre alla collision avoidance sarebbe inoltre opportuno riuscire a garantire anche un'altra proprietà simile: sarebbe desiderabile un meccanismo che vincoli gli agenti a non allontanarsi gli uni dagli altri più di un valore prestabilito.

In molti casi pratici quest'ultima proprietà serve a garantire che la connettività del grafo che sottostà allo sciame sia sempre preservata, Questa caratteristica è nota in letteratura come *connectivity maintenance* o *flocking* nel caso si associ lo sciame di agenti agli sciami che si trovano in natura. In altre parole si desidera che gli agenti possano sempre comunicare gli uni con gli altri direttamente e non.

Visto che quindi si vuole imporre dei vincoli sulle distanze tra gli elementi dello sciame e, considerata la difficoltà di includere all'interno del DF Framework la collision avoidance, è stato sviluppato un metodo alternativo che mantenesse parte dei criteri di ottimalità di cui gode la teoria illustrata finora.

Ferrari Braga in [Ferrari Braga et al., s. d.] risolve il problema della collision avoidance utilizzando la stessa funzione potenziale usata per gli ostacoli. In questo caso però l'equazione del controllo u^{dist} è decisa a tavolino e inoltre utilizza lo stesso termine u^{DF} generato del DF Framework per riuscire a garantire che il vincolo non venga superato.

Invece nel lavoro svolto si è preferito mantenere un approccio più coerente con la teoria fin qui sviluppata: l'obbiettivo dello sciame sarà ancora la minimizzazione di un funzionale di costo e il controllo utilizzato rimarrà a discesa del gradiente.

3.1 Aggiunta di un nuovo funzionale di costo

Per poter garantire che gli agenti non collidano fra loro e che le reciproche distanze non superino mai un determinato valore limite, si è aggiunto un nuovo funzionale di costo J^{dist} al precedente J^{DF} appartenente al DF Framework comprensivo di ostacoli.

In [Stipanović et al., 2007] Stipanović propone un metodo, successivamente applicato da Hussein in [Hussein, Stipanović et al., 2007a], che permette di ottenere collision avoidance fra agenti andando semplicemente a sommare un nuovo funzionale di costo al precedente.

Questa teoria sfrutta sempre il controllo a discesa del gradiente e un funzionale che tende all'infinito qualora due agenti stessero per scontrarsi.

Si definisce quindi il nuovo funzionale di costo da minimizzare come:

$$J = J^{DF} + \omega J^{dist} \geq 0 \tag{3.1}$$

dove $\omega > 0$ è un peso arbitrario.

Considerando proprio J come candidata di Lyapunov, si analizza la sua derivata:

$$\dot{J} = \dot{J}^{DF} + \omega \dot{J}^{dist} = J_t^{DF} + \frac{\partial J^{DF}}{\partial p} \cdot \dot{p} + \omega \frac{\partial J^{dist}}{\partial p} \cdot \dot{p} \tag{3.2}$$

Imponendo adesso un controllo del tipo a discesa del gradiente per entrambi i funzionali si ottiene:

$$\dot{p} = u = u^{DF} + u^{dist}, \text{ con } u^{DF} = -k_1 \frac{\partial J^{DF}}{\partial p} \text{ e } u^{dist} = -k_2 \frac{\partial J^{dist}}{\partial p} \tag{3.3}$$

con $k_1, k_2 > 0$ e si può riscrivere l'equazione (3.2) come:

$$\dot{V} = J_t^{DF} - \frac{1}{k_1} u^{DF} \cdot u - \frac{\omega}{k_2} u^{dist} \cdot u = J_t^{DF} - \left[u^{DF}, u^{dist}\right] W \left[u^{DF}, u^{dist}\right]^T \quad (3.4)$$

dove $J_t^{DF} = \frac{\partial J^{DF}}{\partial t}$ rappresenta la componente tempo variante di J^{DF}. Nel seguito si considererà sempre $J_t^{DF} \leq 0$ in quanto è già stato ampiamente dimostrato nei Capitoli 1 e 2 come esso sia nullo per i task statici e semidefinito negativo per quelli dinamici.

W è una matrice di pesi simmetrica. Nel caso in questione risulta essere pari a:

$$\begin{bmatrix} \frac{1}{k_1} & \frac{1}{2}(\frac{1}{k_1} + \frac{\omega}{k_2}) \\ \frac{1}{2}(\frac{1}{k_1} + \frac{\omega}{k_2}) & \frac{\omega}{k_2} \end{bmatrix} \otimes I_{2\times 2} = \bar{W} \otimes I_{2\times 2}$$

il prodotto di Kronecker per $I_{2\times 2}$ si rende necessario in quanto $u \in \mathbb{R}^2$, ma non modifica affatto i risultati a seguire. Il suo unico effetto è quello di raddoppiare tutti gli autovalori della matrice $\bar{W}$.

Poiché $J_t^{DF} \leq 0$, resta solo da valutare il segno di $\bar{W}$ attraverso i suoi autovalori che risultano essere:

$$\lambda_{1,2} = \frac{\omega k_1 + k_2 \pm \sqrt{2}\sqrt{\omega^2 k_1^2 + k_2^2}}{2k_1 k_2}$$

L'unico modo di avere $\bar{W}$ almeno semidefinita negativa, è di portare l'autovalore negativo a 0, imponendo che $k_2 = \omega k_1$.

Perciò imposto $W \leq 0$ si ottiene $\dot{V} \leq 0$. Quindi, ripetendo le considerazioni già precedentemente esposte in Sezione 1.3.4 e 2.2, il sistema evolverà fino a portarsi in una condizione di minimo locale per il funzionale (3.1) (massimi e selle, non essendo stabili, possono essere evitati con l'aggiunta di rumore sul controllo).

Per essere rigorosi è necessario analizzare cosa succede quando il vettore $\left[u^{DF}, u^{dist}\right]^T \in \ker W$, rendendo di fatto $\dot{V}$ dipendente soltanto dalla componente J_t^{DF}:

$$\ker W\Big|_{k_2=\omega k_1} = \begin{bmatrix} 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{bmatrix} \Longrightarrow u^{DF} = u^{dist} \Longrightarrow u = 0$$

la nuova componente di controllo si annulla con quella calcolata attraverso il DF Framework e quindi l'agente resta fermo sul posto.

Nell'analisi appena svolta si è considerato sostanzialmente un solo agente. Nel caso di uno sciame si otterrà banalmente:

$$\dot{J} = J_t^{DF} - \sum_i \left[u_i^{DF}, u_i^{dist}\right] W_i \left[u_i^{DF}, u_i^{dist}\right]^T.$$

Ciò non comporta alcun cambiamento nelle considerazioni esposte precedentemente riguardo autovalori, stabilità e kernel delle componenti di controllo.

Si noti infine come l'aver assicurato $\dot{J} \leq 0$ non implichi che entrambi i funzionali J^{DF} e J^{dist} debbano sempre decrescere. Come si vedrà negli esempi capita spesso

che J^{dist} aumenti, ma ciò implica che sarà J^{DF} a decrescere. In pratica è ammesso che gli agenti stiano tra loro vicini o lontani quando questo comportamento migliora le prestazioni dello sciame, senza però superare i vincoli.

3.2 Modellazione matematica del vincolo sulle distanze inter-agente

L'analisi svolta nella sezione precedente e quella a seguire derivano sostanzialmente dalla teoria dei potenziali (si veda [LaValle, 2006]) a cui è applicato il controllo a discesa del gradiente ([Bertsekas, 1999], [LaValle, 2006]). Tuttavia essa si differenzia in quanto il potenziale non è fornito esplicitamente, ma è implicito nella forma del funzionale di costo.

Per quanto riguarda l'applicazione al campo dei sistemi multi agente, è possibile trovare soluzioni simili da parte di Hussein e Stipanović in [Hussein, Stipanović et al., 2007a] [Hussein, Stipanović et al., 2007b] e [Hussein, Stipanović, Wang et al., 2007], dove, ad una legge di moto principale (simile alla precedente u_{DF}), si vanno a sommare altri controlli derivati proprio dalla minimizzazione di ulteriori potenziali aggiunti a quello principale.

A differenza però degli studi appena citati, dove ad ogni funzionale corrisponde un determinato vincolo, in questo caso ne è stato sintetizzato uno in grado di garantire che le distanze inter-agenti restino sempre fra un valore massimo e uno minimo prestabiliti, mantenendo tuttavia un buon grado di flessibilità come si vedrà nel seguito.

Per realizzare la collision avoidance e connectivity maintenance fra coppie di agenti si può ricorrere ad un funzionale che vada all'infinito man mano che la distanza fra gli agenti tende a quella massima o minima. L'idea è simile a quella sfruttata nella Sezione 2.3 per realizzare l'obstacle avoidance, e la dimostrazione segue da vicino quella della Proposizione 1.

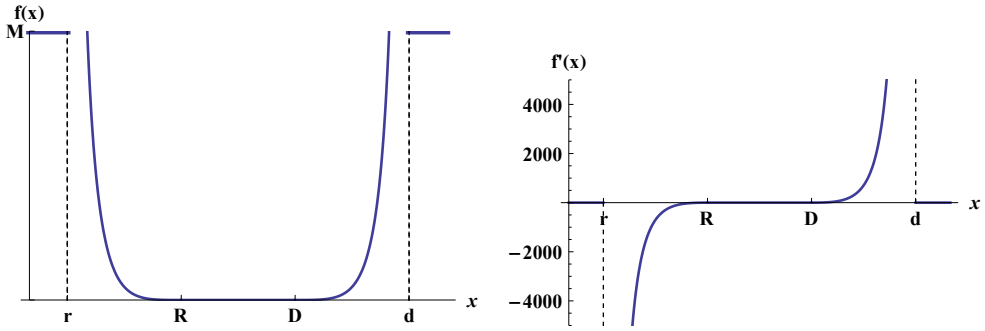
L'espressione analitica del funzionale di costo per i vincoli sulle distanze è la seguente:

$$J^{dist} = \sum_{i,j \neq i}^{N_a} f(\|p_i - p_j\|) \tag{3.5}$$

dove:

$$f(x) = \begin{cases} \max(0, \frac{(R-x)^3(x-D)^3}{(x-r)(x-d)}) & \text{per } r \leq x \leq d \\ M & \text{per } x \leq r \vee x \geq d \end{cases} \tag{3.6}$$

con $0 \leq r < R \leq D < d$, M un valore grande a piacere e p_i vettore delle posizioni dell'i-esimo agente (senza considerare l'orientazione).



(a) $f(x)$ **(b)** $f'(x)$

Figura 3.1: Grafico di $f(\cdot)$ e della sua derivata $f'(\cdot)$.

In Figura 3.1a è illustrata $f(\cdot)$. Si può notare come essa abbia due asintoti verticali, uno su r e uno su d. Il primo permette, come già accennato, di garantire obstacle avoidance con distanza minima inter-agente pari a r. Il secondo invece viene utilizzato per vincolare gli agenti ad avere una distanza, gli uni dagli altri, non superiore a d. Perciò r e d sono rispettivamente le distanze minime e massime consentite.

Un altra caratteristica interessante è data dal fatto che $f(\cdot) = 0$ per $R \leq x \leq D$. Ciò implica che le coppie di agenti che hanno una distanza compresa fra R e D, non danno contributi al funzionale J^{dist}. Questi due parametri aggiuntivi aumentano la flessibilità del funzionale J^{dist} permettendo di sceglierlo ad-hoc a seconda del caso di applicazione.

Infine si noti come $f(\cdot)$ sia stata costruita in maniera tale da essere continua e derivabile con continuità in $]r, d[$. Si può osservare l'andamento della derivata di $f(\cdot)$ in Figura 3.1b.

Nel caso si fosse interessati a imporre soltanto uno dei due vincoli allo sciame è possibile considerare solo le componenti dell'equazione (3.6) di interesse, scartando le altre. Oppure, più semplicemente, si scegliere portare i parametri r, R, d, D fuori scala rispetto all'applicazione che lo sciame deve eseguire.

3.3 Analisi della legge di controllo

Ora che il funzionale di costo J^{dist} è stato scelto analiticamente, è possibile valutare la forma esplicita della legge di controllo u^{dist} da esso derivante.

Per fare ciò, si potrebbe ricorrere direttamente all'equazione (3.3) della Sezione 3.1. Tuttavia si preferisce iniziare dalla derivata del funzionale J^{dist} stesso, per mettere in evidenza come si possa scegliere a colpo d'occhio la forma del controllore:

$$\begin{aligned} \dot{J}^{dist} &= \sum_i \sum_{j \neq i} f'(\|p_i - p_j\|) \frac{p_i - p_j}{\|p_i - p_j\|} \cdot (\dot{p}_i - \dot{p}_j) = \\ &= 2 \sum_i \left[\sum_{j \neq i} f'(\|p_i - p_j\|) \frac{p_i - p_j}{\|p_i - p_j\|} \right] \cdot \dot{p}_i \end{aligned} \tag{3.7}$$

dove la seconda uguaglianza è ottenuta attraverso una semplice manipolazione degli indici.

È quindi evidente la forma che assumerà il controllore:

$$u_i^{dist} = -k_{2,i} \frac{\partial J^{dist}}{\partial p_i} = -k_{2,i} \sum_{j \neq i} f'(\|p_i - p_j\|) \frac{p_i - p_j}{\|p_i - p_j\|}, \text{ con } k_{2,i} > 0 \tag{3.8}$$

Analizzando la struttura della legge di controllo ottenuta si nota che, sull'i-esimo agente, è come se agissero $N_a - 1$ forze, dirette verso ogni altro agente e con modulo e verso decisi da $-k_{2,i} f'(\|p_i - p_j\|)$.

In particolare, come si nota dalla Figura 3.1b, e come è corretto aspettarsi, il controllo spinge l'agente i in direzione esattamente opposta a quella di $\bar{j}$ quando la loro distanza è minore di R. Infatti in questo caso si ottiene $-k_{2,i} \cdot f' > 0$, e quindi gli agenti subiscono una forza repulsiva che li allontana.

Invece u^{dist} muove l'agente i-esimo verso il $\bar{j}$-esimo quando questi superano la distanza D poiché $-k_{2,i} \cdot f' < 0$.

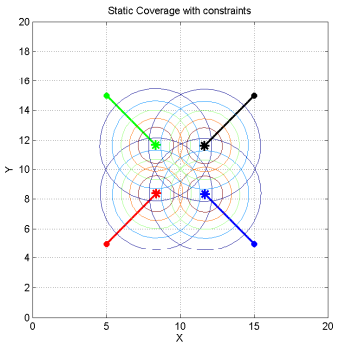
È importante sottolineare come lo stesso identico effetto avvenga contemporaneamente su $\bar{j}$ rispetto a i.

Infine se i e $\bar{j}$ si trovano ad una distanza compresa fra R e D allora il contributo al controllo u_i^{dist} è nullo. Perciò un agente che si trovasse, rispetto a tutti gli altri componenti dello sciame, a distanza comprese fra R e D sarebbe soggetto solamente a u^{DF}, come se i vincoli imposti non esistessero.

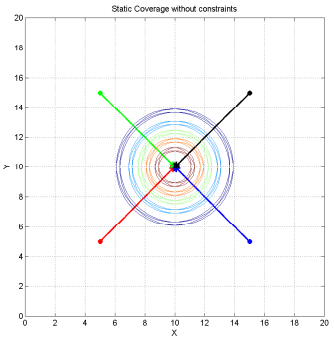
Un ultima considerazione: a differenza di quanto sviluppato per l'obstacle avoidance, questo controllo non tiene assolutamente conto delle DF degli agenti, ma solo delle distanze fra di essi. Ciò è dovuto al fatto che il controllore è ottenuto da un funzionale totalmente indipendente da J^{DF}, che invece si basa sulle funzioni descrittive.

3.4 Conclusioni ed esempi

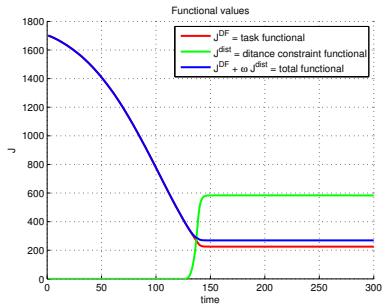
In definitiva all'interno del capitolo è stato introdotto un nuovo funzionale di costo J^{dist} che si somma al precedente J^{DF}. L'aggiunta di questo nuovo elemento garantisce che gli agenti non collidano tra loro e che la distanza massima tra due agenti non sia superiore ad un valore specificato. Coerentemente con quanto sviluppato nel DF Framework la legge di controllo è ancora a discesa del gradiente.



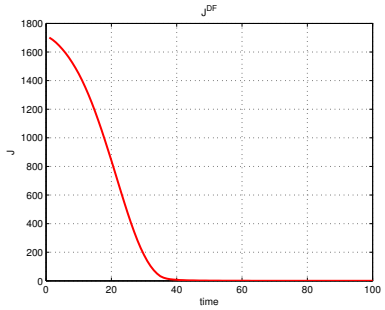
(a) *Percorso effettuato*



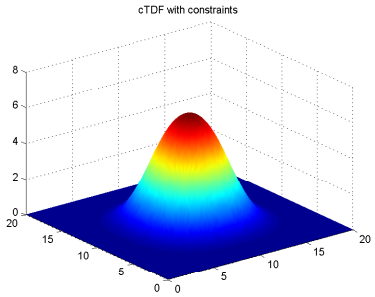
(b) *Percorso effettuato senza vincoli*



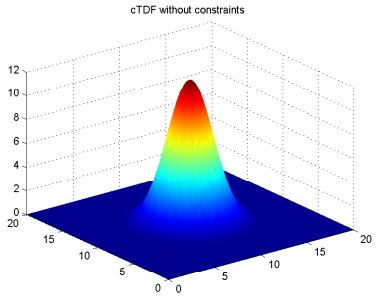
(c) *Funzionali di costo*



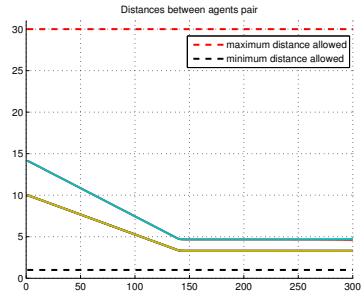
(d) *Funzionale di costo*



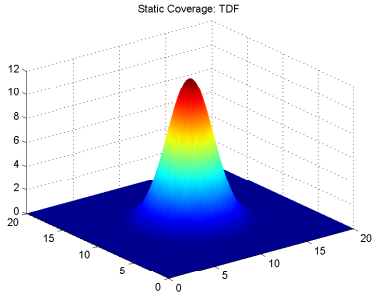
(e) *cTDF con vincoli*



(f) *cTDF senza vincoli*



(g) *Distanze inter-agente*



(h) *TDF*

Figura 3.2: Static Coverage Task svolto con i vincoli sulle distanze (immagini a sinistra) e senza (immagini a destra). Le ultime due vignette mostrano le distanze (che purtroppo risultano sovrapposte) tra gli agenti nel caso di moto vincolato e la forma della TDF utilizzata.

Se l'aggiunta dei vincoli fornisce delle garanzie sul movimento coordinato dello sciame, dall'altro può limitarne le prestazioni, anche se non è sempre detto che ciò accada.

In Figura 3.2 viene confrontato uno di Static Coverage Task svolto da quattro agenti sia nel caso in cui siano attivi i vincoli sulle distanze sia per uno non vincolato.

Si può subito notare come la cTDF degli agenti che non hanno distanze minime o massime (Figura 3.2f) sia molto più simile alla TDF del task rappresentata in Figura 3.2h rispetto al caso opposto di Figura 3.2e.

Di conseguenza i funzionali di costo avranno valori differenti. Infatti nel caso non vincolato il funzionale decresce fin quasi a zero, come si può osservare in Figura 3.2d. Invece nell'altro caso il funzionale complessivo decresce arrivando soltanto a 300.

Ovviamente se gli agenti non fossero puntiformi ma avessero una forma fisica il caso non vincolato li vedrebbe collidere l'uno con l'altro.

Osservando la Figura 3.2g e contemporaneamente l'andamento di J^{dist} in Figura 3.2c si può osservare come il funzionale dapprima nullo aumenti gradualmente man mano che le distanze tra gli agenti si riducono portandosi a valori inferiori a $R = 4$. Tuttavia $J = J^{DF} + \omega J^{dist}$ risulta comunque decrescente seguendo la teoria esposta in Sezione 3.1. Nell'esempio illustrato è stato utilizzato $\omega = 0.075$ al fine di attenuare la reazione del controllo u^{dist}.

Nel caso appena discusso la collision avoidance limita, come è normale aspettarsi, le capacità dello sciame. Tuttavia non è sempre vero che l'aggiunta di vincoli diminuisce le prestazioni della squadra di agenti.

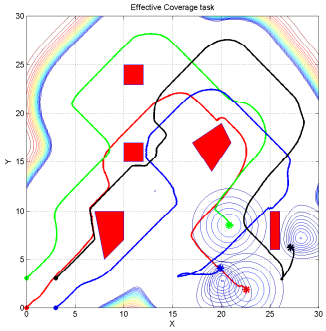
Nella Sezione 2.5.1 si è discusso di come possa accadere che il controllo di un agente risulti nullo anche dopo pochi passi di simulazione: $u^{DF} = 0$ perché la TEF intorno all'agente è nulla.

Aggiungendo adesso il vincolo sulla distanza massima si ottiene che gli agenti che sarebbero dovuti rimanere fermi continuano invece a muoversi seguendo il resto dello sciame che invece si dirige verso la TDF.

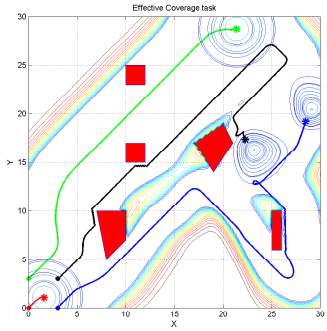
Nelle Figure 3.3a e 3.3b è possibile osservare proprio il comportamento appena descritto. In particolare l'agente rosso resta praticamente fermo nel caso non vincolato, mentre attivando il funzionale J^{dist} esso inizia a muoversi per restare sufficientemente vicino agli altri elementi dello sciame.

In Figura 3.3c è mostrato l'andamento delle componenti di controllo per tutti gli agenti. In alto è rappresentato u^{DF} e in basso u^{dist} divisi sui due assi di moto x e y. Le frecce sovrapposte al grafico indicano una finestra temporale in cui $u^{DF} = 0$ perché la TEF attorno all'agente è nulla. Tuttavia, se questa componente di controllo risulta azzerata, non lo è quella relativa ai vincoli sulle distanze.

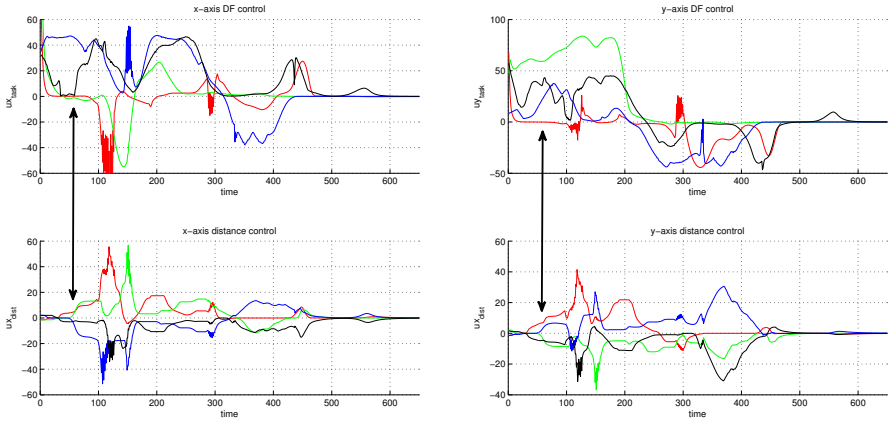
In generale ciò fa si che l'agente rosso, seguendo il resto dello sciame, possa tornare attivo nel suo compito di perlustrare il territorio. Infatti, sempre dalla Figura



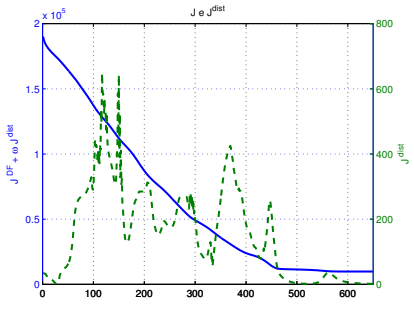
(a) *Percorso effettuato*



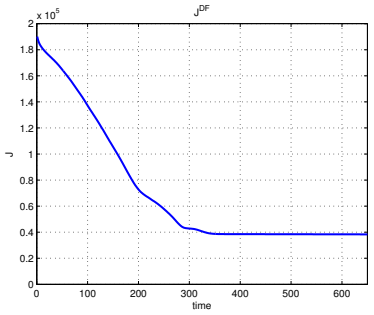
(b) *Percorso effettuato senza vincoli*



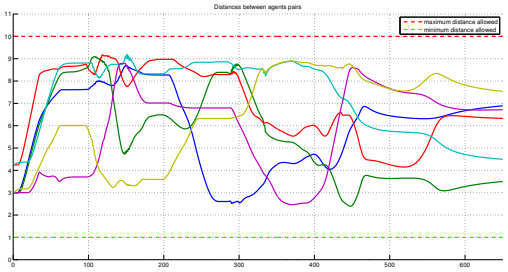
(c) *Azioni di controllo degli agenti con vincoli attivi*



(d) *Funzionale di costo*

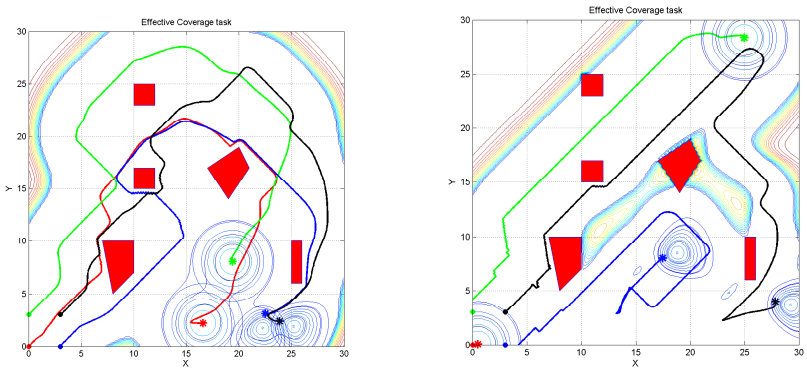


(e) *Funzionale di costo senza vincoli attivi*



(f) *Distanze tra gli agenti nel caso di vincoli attivi*

Figura 3.3: Confronto di un Effective Coverage Task svolto con i vincoli sulle distanze e senza.



(a) *Percorso effettuato senza vincoli di distanza minima*

(b) *Percorso effettuato senza vincoli di distanza massima*

Figura 3.4: Confronto di un Effective Coverage Task svolto con una parte dei vincoli sulle distanze.

3.3c, si nota come u^{DF} ricominci ad avere valori non nulli.

Anche se non è matematicamente dimostrato, questo effetto, dovuto alla distanza massima consentita fra agenti, si ripercuote positivamente anche sul funzionale di costo. Nelle Figura 3.3d è mostrato l'andamento del funzionale complessivo J e, su un altra scala J^{dist}. Invece in Figura 3.3e è mostrato J nel caso non sia considerato alcun vincolo. Confrontandolo con l'altro funzionale si nota la differenza al valore finale: il task effettuato con i vincoli sulle distanze porta a risultati migliori.

Tornando ancora alla Figura 3.3d si osservi come J^{dist} presenti dei picchi proprio quando la componente del controllo u^{dist} è massima, ossia quando un agente è troppo vicino o troppo distante dagli altri.

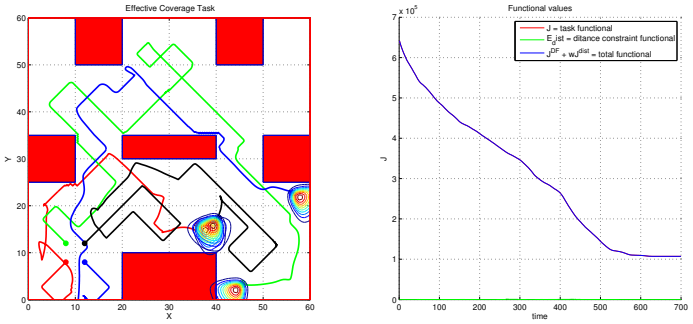
Per completezza in Figura 3.4 sono riportati anche i casi in cui gli agenti siano soggetti soltanto ad un vincolo. In Figura 3.4a si considera solo la connectivity maintenance e difatti il movimento complessivo dello sciame è simile a quello di Figura 3.3a.

Invece in Figura 3.4b è attiva solo la collision avoidance. In questo caso il percorso è molto più simile alla simulazione senza vincoli, tuttavia gli agenti, non potendo stare vicini, ottengono comunque valori di J inferiori.

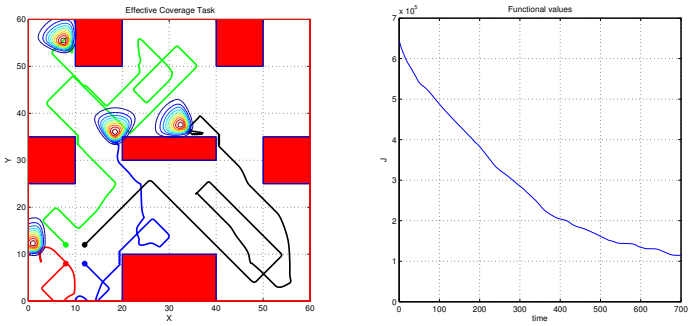
Le considerazioni appena svolte sono tuttavia di carattere generale. Non è detto che un task svolto con i vincoli debba fornire risultati migliori rispetto ad uno in cui non vengono considerati.

In Figura 3.5 è possibile osservare proprio questo fenomeno. Le due simulazioni arrivano ad avere all'incirca lo stesso valore del funzionale J nonostante la prima delle due limiti le distanze tra gli agenti.

Anche in questo caso, dove J raggiunge ottimi valori, si assiste alla "perdita" di



(a) *Simulazione con vincoli*



(b) *Simulazione senza vincoli*

Figura 3.5: Confronto di un Effective Coverage Task comprensivo di vincoli e non.

un agente (in figura risulta essere quello rosso). In generale questo fenomeno non è auspicabile e quindi è sempre bene richiedere una distanza massima fra coppie di agenti.

Capitolo 4

DF Framework con connettività

Nel Capitolo 3 stato sviluppato un metodo per aggiungere condizioni che vincolino le distanze inter-agente, mantenendo tuttavia un approccio a discesa del gradiente di un funzionale di costo.

La legge di controllo ottenuta si compone di due termini: uno che indirizza l'agente verso il task e l'altro che si occupa di portare l'agente in zone dove non vengano violati i vincoli sulle distanze massime e minime.

Non è stata però in alcun modo considerata la formalizzazione iniziale del problema, ossia non è stato tenuto conto dell'esistenza delle Funzioni Descrittive degli agenti, ne di quella del task da portare a termine.

In sostanza il meccanismo utilizzato prescinde dalla base su cui si appoggia il resto del controllo. Se gli agenti fossero stati controllati con altre tecniche o algoritmi, che sfruttassero anch'essi il controllo a discesa del gradiente, si sarebbe potuto condurre il medesimo ragionamento esposto nel Capitolo 3, ottenendo gli stessi risultati.

Ci si chiede allora se sia possibile formalizzare, all'interno del DF Framework, l'imposizione di vincoli sulla distanza fra agenti.

4.1 Connectivity Descriptor Function

Ripensando a come agisce la TEF sugli agenti, ci si accorge che valori positivi hanno l'effetto di attrarre mentre quelli negativi di respingere gli agenti. Viene quindi naturale pensare che, per mantenere lo sciame compatto, si possa aggiungere una Funzione Descrittiva di Connettività $d_{conn}(q, p) > 0$, così da richiamare gli agenti e mantenerli in gruppo.

Un'idea simile è già stata sviluppata in [Ferrari Braga, 2013], tuttavia l'approccio è leggermente differente. Lo studio di Ferrari Braga prevede che la funzione descrittiva di connettività venga moltiplicata alla TEF. In questo modo l'errore cresce in

prossimità di d_{conn} e di conseguenza gli agenti ne sono attratti restando perciò in gruppo.

In questa tesi si prevede invece di sommare la nuova CDF (Connectivity Descriptor Function) all'errore:

$$e(q, pt) = d^*(q, t) + d_{conn}(p, q) - D(p, q) - \sum_{k=1}^{N_{obs}} d_{obs^k}(p, q). \tag{4.1}$$

Anche in questo modo la TEF assume valori maggiori laddove si trova d_{conn}. Di conseguenza gli agenti subiranno una sorta di forza attrattiva verso questa funzione e ciò li porterà a restare uniti fra loro.

La forma di $d_{conn}(p, q)$ presa in considerazione è quella di una gaussiana (equazione (4.2)), in quanto continua e derivabile con continuità, ma nulla vieta di utilizzare le altre DF omnidirezionali presentate nel Capitolo 1:

$$d_{conn} = Ae^{\frac{1}{2}(c(p)-q)^T S^{-1}(c(p)-q)} \tag{4.2}$$

dove $S = diag(\sigma_1, \sigma_2)$. Il termine $c(p)$ invece è il punto su cui è centrata la gaussiana e può essere in generale uno qualsiasi dello spazio Q. Tuttavia risulta più logico farlo coincidere o con la posizione di un agente preposto al mantenimento della connettività dello sciame oppure col baricentro dello sciame stesso.

Sia nella soluzione appena presentata che in quella precedente di Ferrari Braga, gli agenti tendono a restare in gruppo ma non esiste un vincolo di distanza massima. Questo comportamento può non essere del tutto indesiderato in quanto permette a un elemento dello sciame di staccarsi dal gruppo per svolgere un compito a maggiore priorità.

Se invece si desidera porre una distanza limite, tra il centro della gaussiana di connettività e gli agenti, si può applicare il medesimo concetto utilizzato per garantire l'obstacle avoidance, ossia fare in modo che il funzionale J^{DF} tenda all'infinito nel momento in cui il vincolo stia per essere violato.

Infatti se è certo che $\dot{J}^{DF} \leq 0$ e che $J^{DF} \longrightarrow +\infty$ nel caso in cui il vincolo stia per essere superato, allora è matematicamente garantito che nessun agente potrà sforare la distanza massima prevista. Si nota facilmente come l'approccio sia proprio il medesimo della Proposizione 1 della Sezione 2.3. Resta quindi da individuare come si possa far tendere J^{DF} all'infinito utilizzando d_{conn}.

Analizzando la forma scelta per la CDF nell'equazione (4.2) si intuisce come la scelta migliore, per aumentare il valore di J^{DF}, sia quella di far variare l'ampiezza A. Essa deve essere tanto più grande tanto più gli agenti sono distanti da $c(p)$.

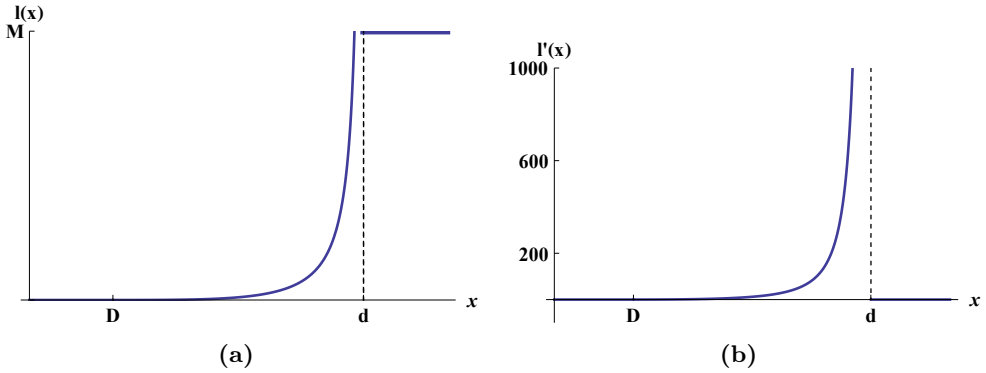


Figura 4.1: Grafico della funzione $l(\cdot)$ e della sua derivata.

4.2 Modellazione matematica della CDF

È stato quindi chiarito come la funzione $d_{conn} \geq 0$, con ampiezza variabile, possa mantenere gli agenti in gruppo, implementando quelli che sono vincoli sulla distanza massima all'interno dello sciame. Resta perciò da scegliere una funzione adatta che faccia variare adeguatamente l'ampiezza della gaussiana.

Affinché $A(p)$ tenga conto di tutti gli agenti si impone che:

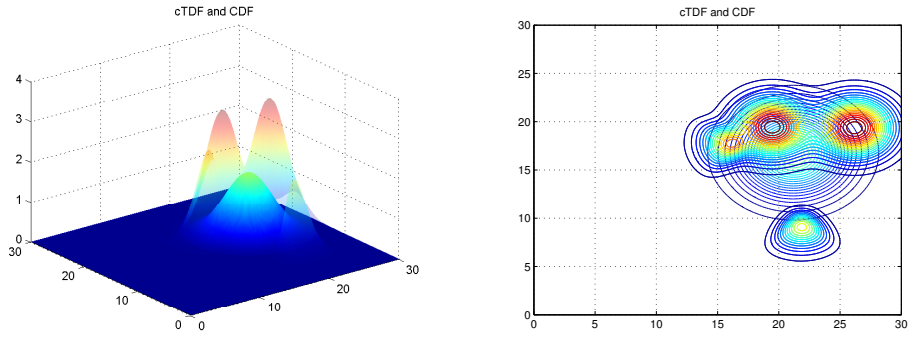
$$A(p) = \sum_{i=1}^{N_a} l(\|p_i - c(p)\|). \tag{4.3}$$

La funzione $l(\cdot)$ è costruita allo stesso modo di quella utilizzata per caratterizzare le ODF (si veda l'equazione (2.7)), semplicemente sono invertite di ruolo R e r. Tuttavia tali parametri verranno chiamati D e d rispettivamente, per coerenza con il lavoro svolto nel Capitolo 3 (D e d erano infatti riferiti al vincolo di connectivity maintenance):

$$l(x) = \begin{cases} \max(0, -\frac{(x-D)^3}{x-d}) & \text{per } x \leq d \\ M & \text{per } x \geq d \end{cases} \tag{4.4}$$

con $0 \leq D < d$. Una rappresentazione di tale funzione e della sua derivata sono mostrate in Figura 4.1.

Risulta quindi che $d_{conn} = 0$ quando le distanze di tutti gli agenti dal punto $c(p)$ sono minori di R. In generale la gaussiana non sarai mai tutta nulla ma non avrà neanche valori troppo elevati. In Figura 4.2 è possibile osservarla mentre mantiene in gruppo quattro agenti. L'immagine si riferisce ad un esempio che verrà esposto più avanti ed in particolare l'istantanea è stata scattata nel momento di massima intensità della CDF rispetto a tutta la simulazione, a conferma che l'ampiezza non cresce poi molto in situazioni standard.



(a) **(b)**

Figura 4.2: DF di connettività e DF degli agenti (in trasparenza) durante una simulazione.

4.3 Analisi della legge di controllo

Adesso che la Funzione Descrittiva di Connettività è stata definita a livello matematico è necessario analizzare il controllore che si ottiene dalla sua implementazione nel DF Framework. La metodologia fortunatamente è molto simile a quella già utilizzata per l'inclusione dell'obstacle avoidance nel Capitolo 2.

Resta tuttavia da precisare esattamente a cosa corrisponda il centro di d_{conn}. Le scelte plausibili sono due, o un agente o il centro di massa dello sciame. In questo caso si è scelto il baricentro poiché da origine ad un controllo più complesso da calcolare e più generale. Modificare tale scelta implica semplificare alcuni termini nella legge di controllo che risulterà dal calcolo del gradiente. È stato deciso in sostanza di affrontare il caso peggiore. Avremo perciò $c(p) = \frac{1}{N_a}\sum_i^{N_a} p_i$, nei calcoli a seguire verrà indicato solo come c.

Per ottenere il controllo u si ricorre come al solito al gradiente del funzionale di costo J. Tuttavia in questo caso, per calcolare correttamente tutte le derivate parziali si preferisce incominciare da $\dot{J}$, analizzando cioè la derivata della candidata di Lyapunov per poi imporne il segno mediante la scelta della legge di controllo:

$$\dot{J} = \int_Q f_e \frac{\mathrm{d}d^*}{\mathrm{d}t} - \sum_i \int_Q f_e \frac{(d_i + \sum_k d_{obs^k})}{\partial p_i} \cdot p_i + \int_Q f_e \frac{\mathrm{d}d_{conn}}{\mathrm{d}t} \tag{4.5}$$

dove $f_e = \frac{\partial f(e)}{\partial e} = 2\max(g, e)$.

Dalla teoria illustrata nei Capitoli 1 e 2 la forma dell'equazione (4.5) è ormai nota per tutti gli addendi tranne che per l'ultimo. Questo infatti è il nuovo termine derivante proprio dall'aver introdotto d_{conn}.

Si analizza perciò l'ultimo addendo di (4.5) indicando con Exp l'espressione $e^{\frac{1}{2}(c-q)^T S^{-1}(c-q)}$:

$$\frac{\mathrm{d}d_{conn}}{\mathrm{d}t} = \dot{A}Exp - AExp(c-q)^T S^{-1}\dot{c},$$
$$\dot{A} = \sum_i^N l'(\|p_i - c\|)\frac{(p_i-c)^T}{\|p_i-c\|}(\dot{p_i} - \dot{c}),$$
$$\dot{c} = \frac{1}{N}\sum_j^N \dot{p_j}$$

Attraverso qualche passaggio algebrico e rimaneggiando gli indici si riesce ad ottenere la seguente forma:

$$\frac{\mathrm{d}d_{conn}}{\mathrm{d}t} = \sum_i \Bigg\{ l'(\|p_i - c\|)\frac{(p_i-c)^T}{\|p_i-c\|} \quad + \\ - \frac{1}{N}\sum_j \Big[l'(\|p_j - c\|)\frac{(p_j-c)^T}{\|p_j-c\|}\Big] \quad + \\ - \frac{A}{N}(c-q)^T S^{-1} \Bigg\} \cdot \dot{p_i} Exp \quad (4.6)$$

Per brevità e per chiarezza si rinominano i tre addendi presenti all'interno delle parentesi graffe come α_1, α_2 e $\alpha_3$rispettivamente. Quindi risulta che:

$$\frac{\mathrm{d}d_{conn}}{\mathrm{d}t} = \sum_i \{\alpha_1 + \alpha_2 + \alpha_3\} Exp \cdot \dot{p_i} = \sum_i \frac{\partial d_{conn}}{\partial p_i} \cdot \dot{p_i}$$

Riferendosi all'equazione (4.5), per garantire che $\dot{J} \leq 0$ bisogna scegliere $\dot{p_i} = u_i = u_i^{task} + u_i^{obs} + u_i^{conn}$. I primi due termini di questa somma sono gli stessi visti nei capitoli precedenti, l'ultimo, nato dall'utilizzo della CDF, risulta essere:

$$u_i^{conn} = -\int_Q f_e \frac{\partial d_{conn}}{\partial p_i}\,\mathrm{d}q = -\int_Q f_e \{\alpha_1 + \alpha_2 + \alpha_3\} Exp\,\mathrm{d}q = \bar{\alpha}_1 + \bar{\alpha}_2 + \bar{\alpha}_3. \quad (4.7)$$

Analizzare questa legge di controllo risulta alquanto complesso, tuttavia è comunque possibile porre alcune considerazioni.

Il termine $\bar{\alpha}_1$ risulta essere:

$$\bar{\alpha}_1 = -l'(\|p_i - c\|)\frac{(p_i-c)}{\|p_i-c\|}\int_Q f_e Exp\,\mathrm{d}q. \quad (4.8)$$

Tale componente del controllo u_i^{conn} ricorda una forza che attira l'agente verso il centro della funzione di connettività in quanto proporzionale al vettore $p_i - c$ ma con modulo negativo (quindi diretta verso c). Si tratta quindi di una componente di richiamo. Invece il secondo termine di u_i^{conn} è pari a:

$$\bar{\alpha}_2 = \frac{1}{N}\int_Q f_e Exp\,\mathrm{d}q \sum_{j=1}^{N_a} l'(\|p_j - c\|)\frac{(p_j-c)}{\|p_j-c\|} \quad (4.9)$$

In questo caso invece si ha una somma di forze. Tuttavia è difficile valutarne direzione e verso.

Infine l'ultimo termine:

$$\bar{\alpha}_3 = \frac{1}{2N_a} \int_Q f_e A(c-q) S^{-1} Exp \, \mathrm{d}q = \frac{1}{2N_a} \int_Q f_e d_{conn}(c-q) S^{-1} \, \mathrm{d}q. \tag{4.10}$$

L'espressione ottenuta risulta indipendente dalla posizione degli agenti ed è attrattiva, ossia spinge gli agenti verso il centro della CDF. Inoltre è interessante notare come questo termine aumenti quanta più TEF è presente nel raggio d'azione di d_{conn}. Questo termine è quindi in qualche modo proporzionale alla componente che muove gli agenti verso il task, solo che tende a trattenere gli agenti all'interno della zona di connettività. In pratica maggiore è la TEF, maggiore sarà la componente di controllo che muove gli agenti e che quindi tende a disperdere lo sciame. Con $\bar{\alpha}_3$ si va a smorzare questo effetto di dispersione.

Infine si può notare come sia $\bar{\alpha}_2$ che $\bar{\alpha}_3$ siano uguali per tutti gli agenti, infatti non compare mai l'indice i all'interno delle equazioni (4.9) e (4.10).

Analizzato per quanto possibile il termine u^{conn} e i suoi effetti sugli agenti bisogna sottolineare che d_{conn}, poiché è presente in f_e, influenza anche u_i^{task} e u_i^{obs}. La stessa cosa accadeva già con le ODF.

Per ultimo si vuole porre l'accento sulla scelta dei termini σ_x e σ_y presenti in S^{-1}. È infatti importante che d_{conn}, anche a piccole ampiezze, possa "raggiungere" gli agenti, ossia per ogni agente i si vorrebbe che $d_i(p_i, q) \gg 0$ su quelle $q \in Q$ per cui $d_{conn}(c, q) \gg 0$. In pratica d_{conn} deve essere, di per sé, abbastanza larga altrimenti l'ampiezza della gaussiana potrebbe arrivare ad avere valori troppo elevati. Una buona scelta può essere $\sigma_x = \sigma_y = \sqrt{d}$.

4.4 Conclusioni ed esempi

In questo capitolo si è riusciti ad includere all'interno del DF Framework la connectivity maintenance, che assicura che gli agenti restino sempre in gruppo, o meglio, che non distino dal centro di massa dello sciame più di d.

Il vincolo sulla distanza massima è stato garantito matematicamente in modo simile a quanto fatto per l'obstacle avoidance.

Tuttavia tra le due soluzioni esiste una certa differenza concettuale. L'aver introdotto gli ostacoli a livello di Funzioni Descrittive ha permesso infatti di ottenere un comportamento degli agenti molto particolare: essi non solo schivano l'ostacolo, ma cercano di sprecare meno risorse del payload possibili sulla zona da evitare, in quanto considerata priva di interesse.

L'introduzione della connettività a livello di Framework invece non porta con sé tutte queste considerazioni. Semplicemente lo sciame è portato a restare compatto

all'interno di d_{conn}. L'unico aspetto da non sottovalutare può essere dato dal fatto che gli agenti tendano a rivolgere la loro ADF all'interno dello sciame quando la TDF attorno a loro è nulla. È come se gli agenti senza un task nelle vicinanze si premurassero di tenere d'occhio il resto del gruppo.

Rispetto invece al vincolo sulla distanza massima realizzato nel Capitolo 3 la più grossa differenza sta nel fatto che in questo caso le distanze non sono tra coppie di agenti, ma tra gli agenti e il centro dello sciame. I due controlli inoltre sono concettualmente diversi: uno si innesta alla base del Framework estendendone le possibilità, l'altro invece si affianca al Framework senza considerare nulla della teoria che ne sta alla base. Tuttavia c'è da sottolineare come il controllo sviluppato nel Capitolo 3 risulti assolutamente più semplice sia da analizzare sia a livello computazionale. Infatti le equazioni (4.8), (4.9) e (4.10), contenendo degli integrali di superficie, necessitano di una buona potenza di calcolo.

Per meglio confrontare le due soluzioni proposte in merito alla connectivity maintenance si ripresenta lo scenario analizzato in Sezione 2.5 sfruttando questa volta d_{conn} per assicurare i vincoli.

In Figura 4.3a è possibile osservare il percorso svolto dagli agenti. Con la linea azzurra è stato marcato il movimento di d_{conn}, ossia il baricentro dello sciame. Il risultato è molto simile a quello ottenuto con il controllo esterno u_{dist} di Figura 3.4a.

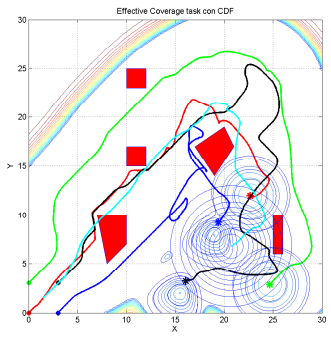
Il valore del funzionale di costo mostra tuttavia una migliore riuscita del task di effective coverage, ottenendo un valori pari a circa 0.2×10^5. Più o meno la metà di quello ottenuto nel Capitolo 3.

Anche in questo caso si può osservare in Figura 4.3c come la nuova componente di controllo u^{conn} permetta all'agente rosso di continuare a muoversi e a partecipare al task. Infatti sia u^{task} che u^{obs} risultano nulle.

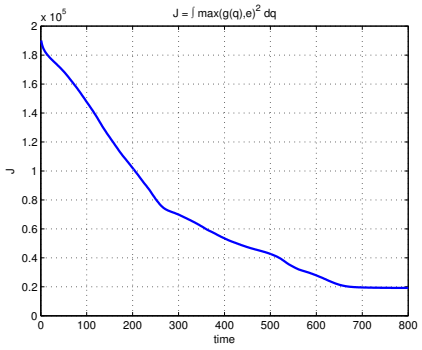
Infine osservando anche le distanze tra agenti e il loro centro di massa (Figura 4.3d) e la conseguente ampiezza della CDF (Figura 4.3e) si può notare che fra gli istanti 100 e 300 tutti gli agenti superano la soglia D e di conseguenza l'ampiezza della gaussiana tende a salire.

Il picco però lo si nota tra gli istanti 400 e 500 in cui l'agente nero si allontana particolarmente dal gruppo. Infatti anche in Figura 4.3c si osserva un rapido aumento di u^{conn} sull'asse y.

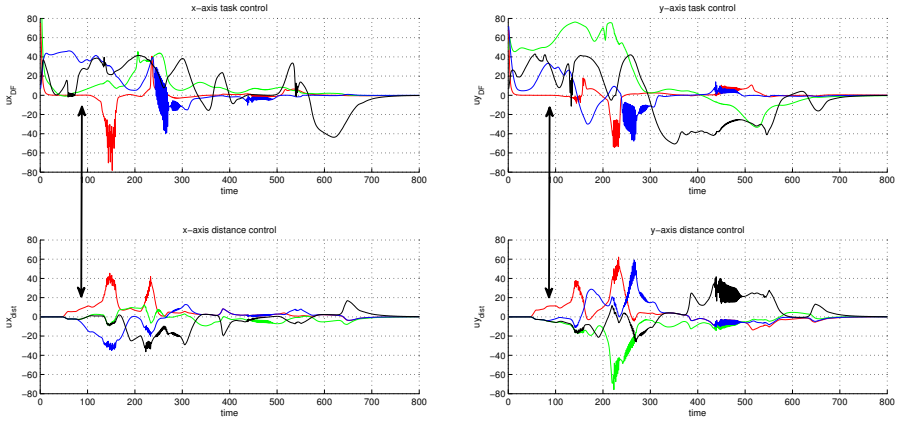
Il fatto che sia il controllo sia l'ampiezza della CDF risultino molto frastagliate e poco morbide è da attribuirsi al passo di campionamento utilizzato nelle simulazioni. Riducendolo si sarebbero ottenute curve più dolci al prezzo di un aumento proporzionale del tempo di simulazione.



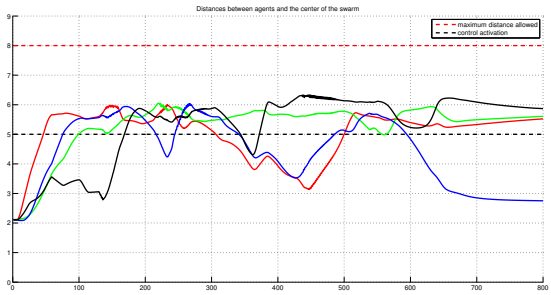
(a) *Percorso effettuato*



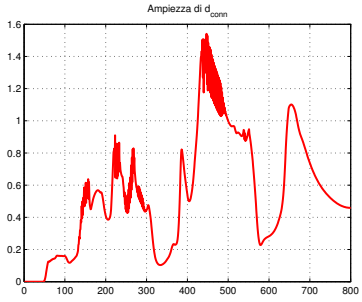
(b) *Funzionale di costo*



(c) *Componenti di controllo*



(d) *Distanze fra gli agenti e il loro centro di massa*



(e) *Ampiezza di d_{conn} nel tempo*

Figura 4.3: Effective Coverage Task con vincoli sulla distanza massima garantiti tramite CDF

Capitolo 5

Intruder Tasks

Il Descriptor Functions Framework è una tecnica che permette ad un insieme di agenti eterogenei di muoversi in maniera coordinata al fine di eseguire un task. Ogni agente, considerato puntiforme e con dinamica a singolo integratore, possiede una Funzione Descrittiva detta ADF. Questa modella matematicamente le caratteristiche, o le proprietà, del payload montato sull'agente. Anche il task è rappresentato da una funzione sparsa sull'area di lavoro e valori elevati indicano un alta richiesta di risorse.

Il Framework, utilizzando un controllo a discesa del gradiente, minimizza un funzionale di costo proporzionale alla differenza fra la funzione del task, detta TDF, e la somma delle funzioni degli agenti. In questo modo gli agenti si muovono verso le zone dello spazio di lavoro dove è richiesta maggior presenza di risorse, identificate nel payload degli agenti.

Nei Capitoli 2 e 4 si è visto come sia possibile includere, a livello di Framework e di Funzioni Descrittive, le proprietà di obstacle avoidance e connectivity maintenance. Queste due caratteristiche mantengono inoltre il concetto di base del Framework: gli agenti evitano gli ostacoli per non sprecare le risorse del payload su un target di poco interesse, mentre invece non si distanziano eccessivamente gli uni dagli altri perché tramite la ADF devono controllore anche cosa stia facendo il resto del gruppo, proprio come se fosse un vero sciame di animali.

5.1 Intruder tracking task

Una delle potenzialità del DF Framework è quella di poter eseguire task differenti senza cambiare legge di controllo. L'unico elemento che deve essere modificato è proprio la Task Descriptor Function. I task sviluppati in [Niccolini, 2011] per il Framework sono molteplici e si possono dividere in generale in due macro categorie: quelli statici (*uniform* e *static deployment*) e quelli dinamici, ossia tempo varianti (*effective* e *dynamic coverage*). I task sopracitati hanno in comune la caratteristica di

essere definiti su tutto il dominio Q_{free}. Questo avviene perché vogliono modellare un'*informazione*, di qualsiasi tipo essa sia, collocata su tutto lo spazio operativo degli agenti.

Si può però pensare a una TDF non nulla solo in una porzione di Q_{free} e che sia, senza perdere di generalità, tempo variante. Verso di essa si muoveranno gli agenti più vicini, quelli per i quali, in termini matematici, $f_e \frac{\partial d_i}{\partial p_i} \neq 0$ che significa che il sensore dell'agente è in grado di rilevare il target. In questo Capitolo perciò si considererà l'agente dotato di un sensore piuttosto che di una risorsa di payload da distribuire in determinati punti dello spazio.

Un esempio pratico del task appena esposto può essere rappresentato da un intruso o comunque da un elemento estraneo al gruppo di agenti che si muove autonomamente nello spazio Q.

Tale elemento può essere considerato puntiforme, proprio come si fa per gli agenti. Inoltre si può porre una Funzione Descrittiva su di esso ottenendo, di fatto, una funzione target $d^*(q,t) > 0$ solo per un'area limitata dello spazio operativo.

La DF posta sull'intruso può essere interpretata in vari modi:

- può rappresentare una densità di probabilità rispetto alla posizione esatta dell'elemento estraneo;
- può descriverne un'approssimazione della forma fisica;
- può rappresentare, se si utilizza una Funzione Descrittiva con FoV, una porzione dell'intruso a maggior interesse per gli agenti.

È stato più volte ribadito, durante i capitoli precedenti, come la legge di controllo tenda in generale a muovere ogni singolo agente verso la zona in cui l'errore e, ossia la d^*, ha valori positivi maggiori. Anche se adesso il target è una funzione che si sposta nel tempo, il comportamento complessivo dello sciame non cambia: gli agenti si muoveranno verso la zona in cui d^* è maggiore, inseguendo di fatto l'intruso.

In [Pallottino, Nardi et al., 2014] e [Pallottino, Caiti et al., 2015] è proposto un approccio simile a quello appena esposto. Degli agenti con dei sensori con FoV sono posti all'interno di un ambiente discretizzato. Quando l'intruso si muove all'interno del cono di vista gli agenti si muovono per convergere su di lui.

Per quanto riguarda il funzionale di costo, non è detto che esso decresca sempre, proprio come nel dynamic coverage task. Tuttavia questo non modifica il modo di agire degli agenti, essi tenderanno sempre a muoversi verso d^* quando risultano sufficientemente vicini.

Indicando con r la posizione dell'intruso e considerando un ambiente sia privo di ostacoli che di vincoli sulle distanze è possibile riscrivere la derivata del funzionale di

costo come:

$$\begin{aligned}\dot{J} &= \int_Q f_e \frac{\partial d^*(q,p)}{\partial r}\,\mathrm{d}q \cdot \dot{r} - \sum_i \int_Q f_e \frac{\partial d_i(q,p)}{\partial p_i}\,\mathrm{d}q \cdot \dot{p} = \\ &= \int_Q f_e \frac{\partial d^*(q,p)}{\partial r}\,\mathrm{d}q \cdot \dot{r} - \sum_i \|u_i\|^2\end{aligned} \tag{5.1}$$

l'ultimo passaggio si ottiene ricordando che la legge di controllo degli agenti è della forma $u_i = \int_Q f_e \frac{\partial d_i}{\partial p_i}$. Come al solito $f_e = \frac{\partial f(e)}{\partial e}$.

Il movimento dell'intruso non è in generale noto a priori ed è perciò difficile dire come possa evolvere il funzionale di costo. Sicuramente, sarà più elevato fintanto che gli agenti sono distanti dal target. Invece decrescerà man mano che gli agenti si avvicinano r, al limite diventando nullo.

La dinamica dell'intruso può essere scelta in vari modi a seconda dello scenario che si sta studiando. In alcuni casi potrebbe essere preferibile un movimento predeterminato, in altri uno completamente randomico. Tra queste due dinamiche esistono tutta una serie di tecniche "smart" che muovono l'intruso in maniera che fugga dagli agenti. Ad esempio si potrebbero utilizzare delle funzioni potenziali che lo allontanino dagli inseguitori come in [Sastry, Antoniades et al., 2003].

È tuttavia interessante analizzare il comportamento del sistema agenti-intruso nel caso particolare in cui anche quest'ultimo abbia una legge di controllo simile a quella degli agenti, ossia:

$$u_r = \dot{r} = \int_Q f_e \frac{\partial d^*}{\partial r} \tag{5.2}$$

La legge appena descritta risulta muovere l'intruso lontano dagli agenti, cercando di evitarli il più possibile. La prova analitica di quanto appena affermato non è evidente in questa espressione, ma lo sarà nella prossima equazione (5.3).

Assegnato quindi il movimento (5.2) all'intruso otteniamo la seguente espressione per la derivata del funzionale di costo:

$$\dot{J} = \|u_r\|^2 - \sum_i \|u_i\|^2 \tag{5.3}$$

J aumenterà o decrescerà a seconda che lo scarto fra l'energia dell'intruso e quella di tutti gli agenti sia positiva o negativa rispettivamente.

L'equazione (5.3) mostra chiaramente come il sistema formi un gioco in cui l'intruso cerca di aumentare il funzionale di costo, mentre gli agenti, in modo cooperativo, cercano di farlo decrescere. Si capisce inoltre come il movimento di r non possa essere altro che quello di "scappare" dagli agenti, in quanto unico modo di aumentare J.

Ovviamente solo gli agenti sufficientemente vicini a $d^*(r,q)$ ne risentiranno la presenza e inizieranno a muoversi verso di esso. Tutti gli altri rimarranno fermi e il

loro contributo a $\dot{J}$ sarà nullo fintanto che l'intruso non si porterà in una posizione sensibile.

Si è così ottenuto un gioco differenziale (si veda [Isaacs, 1954]) di tipo "pursuit-evasion" ([Bryson et al., 1965], [LaValle, 2006][1]) multi agente con sensitività limitata ([Bullo, Bopardikar et al., 2007] [LaValle, 2006][2], [Sastry, Kim et al., 2001]). Infatti le dinamiche, degli agenti e dell'intruso, sono dipendenti dalla posizione di tutte le parti in gioco e cercano ognuna di massimizzare/minimizzare una *payoff function*, ruolo svolto proprio dal funzionale J, come è chiaro dall'equazione (5.3).

Un particolare da notare riguarda l'interazione dell'intruso con il bordo di Q. Come già spiegato nella Sezione 1.3.5 del Capitolo 1, gli agenti tenderebbero ad uscire da Q se si trovassero nelle vicinanze del suo bordo. Tuttavia questo effetto viene evitato grazie alla presenza della funzione max() nell'espressione di $f(e)$. L'intruso invece presenta un comportamento diametralmente opposto: si muove allontanandosi dal bordo, portandosi all'interno di Q.

Il motivo è evidente dalle equazioni (5.2) e (5.3): se parte della DF dell'intruso stesse fuori dall'area di lavoro allora il modulo di u_r decrescerebbe, poiché una frazione di $\frac{\partial d^*}{\partial r}$ non prenderebbe parte al calcolo dell'integrale su Q (si veda l'equazione (5.2)). Di conseguenza anche $\dot{J}$ decrescerebbe in quanto il temine $\|u_r\|^2$, dell'equazione (5.3), è diminuito. Il risultato di tutto ciò è che l'intruso viene come "respinto" dai bordi.

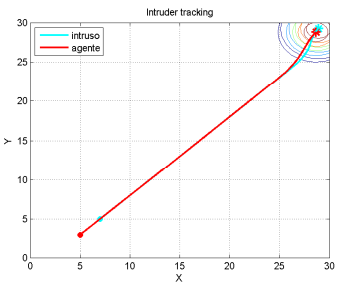
Vista la natura di gioco differenziale del task in esame, risulta naturale chiedersi quali siano le azioni di controllo massime delle parti in gioco e come ne vari l'intensità.

Il valore massimo del controllo non si ottiene, a dispetto di quello che si potrebbe pensare, quando agente e intruso sono sovrapposti, come ad esempio avviene in alcuni scenari di intercettazione tramite PNG, anch'essi descrivibili in termini di gioco differenziale. Infatti man mano che le DF di agente e intruso si intersecano, u_i e u_r aumentano, tuttavia continuando a sovrapporsi si ha che f_e diminuisce. Di conseguenza si ha un istante in cui il controllo raggiunge il valore massimo per poi tornare a decrescere.

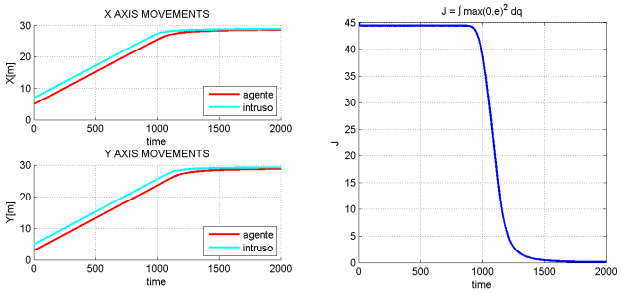
In generale non è facile precisare quale possa essere l'intensità massima dell'azione di controllo. Però, si può sicuramente affermare che maggiore è $\int_Q d_i$ (o $\int_Q d_r$), maggiore sarà u_i^{max}. Perciò se si considera il caso di un solo agente con DF dello stesso tipo dell'intruso, ma di ampiezza inferiore, allora avremo che l'agente non riuscirà a "intrappolare" l'intruso, neanche quando quest'ultimo sarà costretto a rallentare o virare a causa dell'interazione coi bordi. Ovviamente sono da escludere tutti quei casi di simmetria perfetta che bloccherebbero l'intruso.

Questa particolarità non è comunque da considerarsi negativa o insoddisfacente. Infatti il task prevede un inseguimento, un *tracking*, dell'intruso. Non è richiesto che

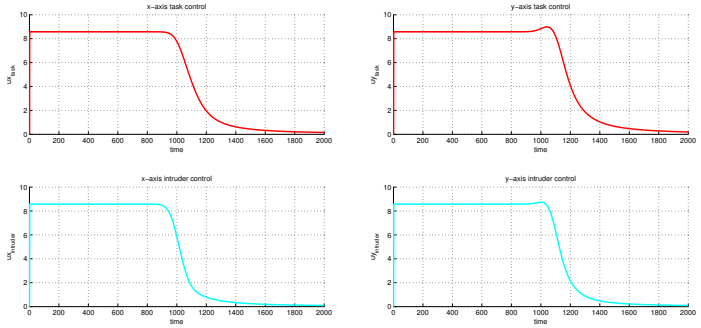
[1] Capitolo 13
[2] Capitolo 12



(a) *Percorso effettuato.*



(b) *Coordinate del movimento e funzionale di costo.*



(c) *Intensità del controllo di agente e intruso.*

Figura 5.1: Gioco a due paritario: le DF di intruso e agente sono identiche.

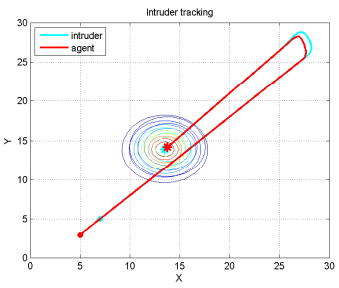
questo venga catturato, bloccato, o intrappolato dagli agenti.

Infine si vuole sottolineare nuovamente come la dinamica dell'intruso scelta nell'espressione (5.2), non sia altro che un caso particolare fra tutte le dinamiche possibili. Fintato che la sua DF rappresenta d^*, si è sicuri che gli agenti di muoveranno verso di esso, a prescindere dalla strategia utilizzata dall'intruso. La particolare scelta proposta ha permesso però di effettuare un'interessante analisi, mettendo alla pari agenti e intruso e realizzando un gioco multi-agente.

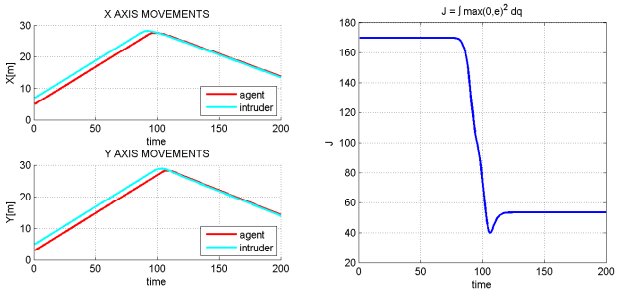
5.2 Commenti ed esempi

Per validare l'analisi condotta nella sezione precedente si propongono, a seguire, una serie di esempi in cui uno o più agenti inseguono l'intruso controllato attraverso l'equazione (5.2).

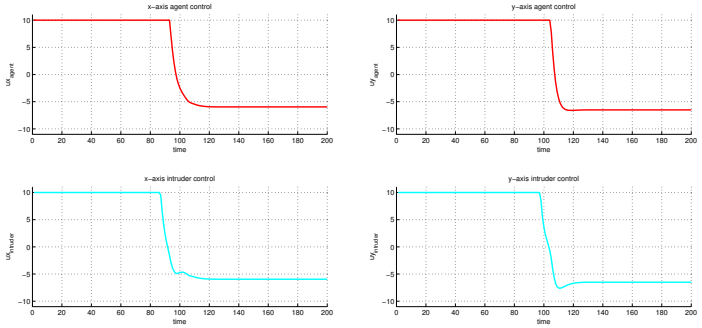
In Figura 5.1 si può osservare un caso di gioco a due del tutto paritario, l'agente e l'intruso hanno la stessa Funzione Descrittiva con gli stessi parametri. Quello che si vuole evidenziare è come le azioni di controllo di entrambi siano uguali e restino costanti fino al momento in cui l'intruso è obbligato a cambiare la sua strategia a causa del bordo che, come già detto, ha un effetto "repulsivo".



(a) *Percorso effettuato.*



(b) *Coordinate del movimento e funzionale di costo.*



(c) *Intensità del controllo di agente e intruso.*

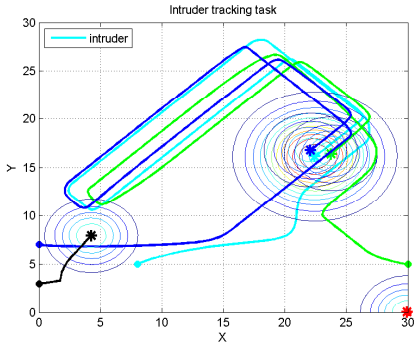
Figura 5.2: Gioco a due impari: la DF dell'intruso è maggiore di quella dell'agente.

È il tipico caso di equilibrio di Nash in cui nessuna delle parti in gioco, cambiando azione singolarmente, migliora la sua situazione. Appena l'intruso è costretto a variare il controllo, il funzionale decresce a significare un vantaggio dell'agente. Poiché a fine simulazione $J = 0$ allora significa che l'agente è riuscito a bloccare l'intruso.

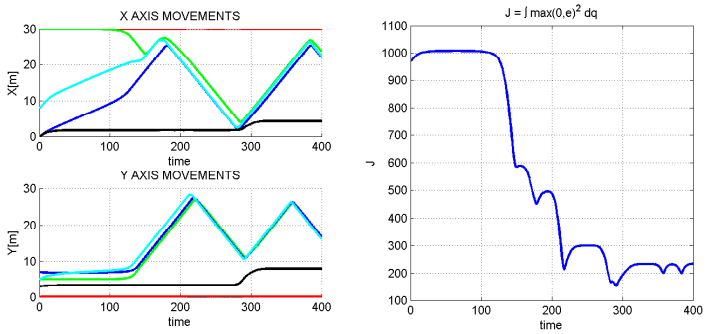
Nell'esempio di Figura 5.2 l'intruso ha DF con ampiezza maggiore. Questo gli consente di avere un valore massimo di controllo più elevato rispetto all'agente e quindi non viene messo alle strette, come nel caso precedente, ma continua a muoversi. Il sistema si riporta su un altro punto di equilibrio che risulta tuttavia più favorevole all'agente.

Continuando all'infinito si otterrebbe un funzionale che scende di equilibrio in equilibrio. Non potrebbe tuttavia arrivare a zero in quanto $d_r > d_i$. Questo non vuol dire che l'agente non potrà mai bloccare l'intruso, semplicemente questo accadrà J non sarà nullo. L'agente si avvicina sempre più all'intruso ogni volta che questo è costretto a deviare la sua traiettoria a causa dei bordi. Una volta che i due elementi sono sovrapposti il controllo di entrambi risulterà nullo. In Figura 5.2a e 5.2b si può già notare quanto l'agente risulti vicino all'intruso.

L'esempio riportato in Figura 5.3 considera il caso in cui sono presenti più agenti. Si può osservare come alcuni di questi non prendano parte al tracking o si muovano



(a) *Percorso effettuato.*



(b) *Coordinate del movimento e funzionale di costo.*

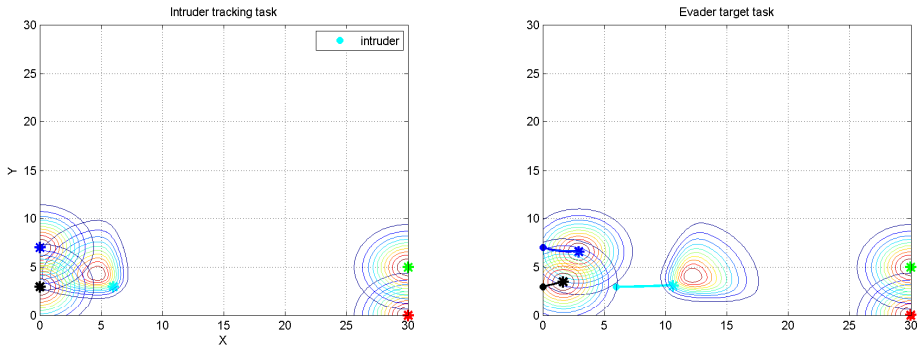
Figura 5.3: Intruder Tracking Task.

solo per un breve tratto. Ciò accade perché l'intruso non è sufficientemente "visibile" dagli agenti. In termini di DF Framework si ha che $f_e = 0$ nelle zone di Q in cui $d_i > 0$ e viceversa, in pratica la DF degli agenti rimasti fermi non ha mai intersecato a sufficienza quella dell'intruso.

Infine in Figura 5.4 è stata utilizzata una Funzione Descrittiva con FoV per l'intruso. Gli agenti partono dalla stessa posizione della simulazione precedente e posseggono le stesse ADF.

In questo caso si osserva come solo gli agenti vicini all'intruso si muovano. Tuttavia quest'ultimo, oltre che allontanarsi dagli agenti, ruota, così da diminuire il più possibile il contatto fra le Funzioni Descrittive. Se si considera la DF dell'intruso come la parte di interesse per lo sciame, allora si può leggere il risultato ottenuto come un movimento atto a "nascondere" questa zona sensibile dai sensori degli agenti. Man mano che l'intruso si sposta e ruota, l'intersezione fra le DF, che rende possibile il movimento degli agenti, diminuisce. Questi, a fine simulazione, non vedono più l'intruso che, ormai ruotato completamente, si ferma.

Si vuole infine sottolineare un dettaglio importante. I task sviluppati nei capitoli precedenti modellano un'informazione, sparsa sull'area di lavoro, del tutto virtuale, fittizia. In particolare ciò che l'agente percepisce con il sensore di payload va a



(a) *Posizione iniziale agenti ed intruso.* **(b)** *Percorso effettuato.*

Figura 5.4: Intruder Tracking Task descritto da DF con *fov*.

vantaggio di terze parti, non viene cioè utilizzata direttamente dal robot, o per lo meno non ne influenza la legge di controllo. Invece nell'intruder task l'agente sfrutta quello che ottiene dai sensori per muoversi. Si pensi ad esempio ad un veicolo autonomo dotato di telecamera. Quando l'intruso compare nelle acquisizioni della camera allora la ADF dell'agente starà intersecando quella dell'intruso. Più a fuoco e più nitido sarà l'intruso più le due DF saranno sovrapposte.

5.3 Intruder searching task

Il task analizzato nelle sezioni precedenti prevede che gli agenti inseguano un intruso. Si è discusso più volte del problema che vede gli agenti restare fermi se l'intruso non è visibile già a inizio scenario. Anche l'intruso a sua volta rimane fermo se viene controllato con la legge dell'equazione (5.2).

Ci si chiede allora se non sia possibile fare in modo che gli agenti cerchino l'intruso sul territorio, ed una volta trovato, lo inseguano. In [Sastry, Kim et al., 2001] e [Sastry, Antoniades et al., 2003] Sastry propone di porre sull'area di ricerca una densità di probabilità che viene man mano portata a zero dagli agenti fino a quando non riescono a identificare l'intruso.

Una prima idea nasce quindi dalla considerazione che un task di ricerca è già stato sviluppato: l'effective coverage. Si può perciò pensare di iniziare con un task di questo tipo e una volta individuato l'obbiettivo, cambiare il task per inseguire l'intruso.

Tuttavia si pone il problema di come definire matematicamente il termine *individuare*. Esistono più scelte possibili, ad esempio l'agente potrebbe distare meno di un valore prefissato dall'intruso, oppure quest'ultimo potrebbe trovarsi sotto la ADF per essere dichiarato come riconosciuto. O ancora, si potrebbe pensare che l'intersezione fra le DF debba superare una certa soglia.

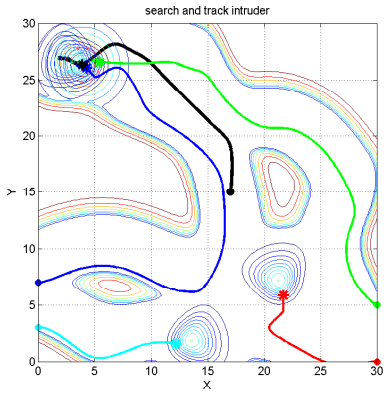


Figura 5.5: Esempio di search intruder task, l'intruso (linea nera) è inizialmente posizionato lontano dagli agenti dotati di FoV.

Fortunatamente si riesce superare questo problema di definizione ricorrendo ad una seconda idea: riunire entrambi i task, effective coverage e intruder tracking, in uno solo invece che passare dall'uno all'altro quando una serie di condizioni sono verificate.

Detta d_{eff} la funzione target dell'effective coverage (si veda l'equazione (1.2.3)), e d_{intr} la funzione obbiettivo dell'intruso, nonché funzione descrittiva posta sopra di esso, è possibile definire la nuova funzione target come:

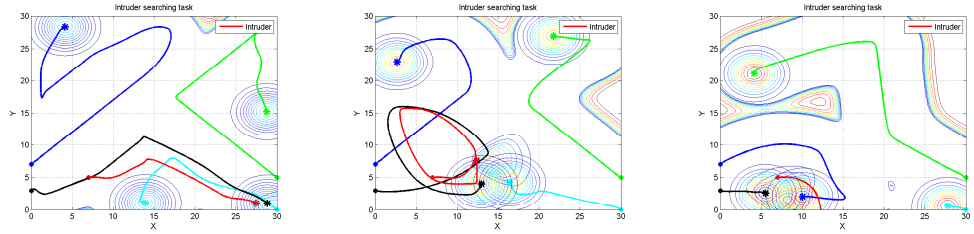
$$d^*_{search} = \max_Q(d_{eff}, d_{intr}) \tag{5.4}$$

d_{eff} e d_{intr} seguono la loro propria dinamica indipendentemente l'una dall'altra.

Quello che si ottiene è uno spazio di ricerca Q che viene man mano esplorato. Istante per istante d_{eff} cala laddove sono presenti le ADF degli agenti e d^*_{search} segue proprio questo andamento tranne dove è situato l'intruso. Lì, il valore della funzione obbiettivo risulterà uguale a quello di d_{intr} se d_{eff} è calato o è nullo, altrimenti sarà il massimo fra le due funzioni.

Con questa modifica se l'intruso si trova in una zona esplorata dagli agenti viene automaticamente intercettato e inseguito. Altrimenti, la ricerca andrà avanti normalmente verso le zone ancora da perlustrare. Un esempio di questo nuovo task è presentato in Figura 5.5: l'intruso, posto in mezzo a Q, viene inizialmente cercato dagli agenti. Una volta identificato, viene inseguito (nel caso in figura dall'agente blu). Gli altri agenti continuano la ricerca, ma quando si trovano nelle vicinanze di r iniziano anche loro il tracking (un esempio è l'agente verde).

Un particolare interessante, se si sfrutta ancora l'equazione (5.2), è dato dal fatto che, se da un lato l'intruso è "respinto" dagli agenti, dall'altro è attratto dalle aree in cui d^*_{search} è positivo. Ciò implica che l'intruso tende a muoversi verso aree di Q ancora inesplorate. Se, ad esempio, un agente, spostandosi, diminuisce il valore



(a) *Percorso effettuato con* $\max(d_{eff}) > \max(d_{intr})$.

(b) *Percorso effettuato con* $\max(d_{eff}) = \max(d_{intr})$.

(c) *Percorso effettuato con* $\max(d_{eff}) < \max(d_{intr})$.

Figura 5.6: Intruder searching task con ampiezze diverse fra d_{eff} e d_{intr}.

di d^*_{search} vicino all'intruso, quest'ultimo tenderà a muoversi in direzione opposta, tornando a "nascondere" la propria DF sotto d_{eff}.

È tuttavia doveroso porre l'attenzione su un aspetto non troppo evidente: bisogna scegliere accuratamente l'ampiezza massima di d_{eff} relativamente a d_{intr}. È questa infatti l'ago della bilancia fra un task e l'altro.

In Figura 5.6 è possibile osservare lo stesso scenario di Figura 5.4, con diverse ampiezze per d_{eff} e d_{intr}. Nello scenario precedente, gli agenti si fermavano senza aver inseguito l'intruso in quanto questo, ruotando, si era reso non osservabile. Con il nuovo task, gli agenti non restano immobili ma si muovono setacciando lo spazio Q e rintracciando quindi l'intruso.

Se $\max(d_{eff}) > \max(d_{intr})$ si otterranno agenti più portati ad esplorare lo spazio Q che a inseguire un eventuale intruso (a meno che d^*_{search} non sia tutta nulla nei dintorni di quest'ultimo). Il motivo è sempre il solito, gli agenti si muovono verso zone in cui la funzione obbiettivo è maggiore. Nell'esempio mostrato in Figura 5.6a l'agente blu, che nella simulazione precedente era quello che eseguiva meglio il task, ora si muove in direzione totalmente differente. Emblematico è il caso dell'agente azzurro che non si cura affatto dell'intruso che invece viene inseguito solo dall'agente nero in quanto attorno ad esso d_{eff} è ormai tutta nulla.

Porre invece $\max(d_{intr}) \geq \max(d_{eff})$ significa dare maggiore priorità al task di tracking piuttosto che a quello di ricerca. Gli agenti si muoveranno nello spazio, ma una volta scorto l'intruso inizieranno a seguirlo. In Figura 5.6b e 5.6c si può notare la differenza di comportamento fra i casi $\max(d_{intr}) = \max(d_{eff})$ e $\max(d_{intr}) > \max(d_{eff})$ rispettivamente. Nel primo solo l'agente nero individua l'intruso e lo insegue (a fine simulazione si aggiunge anche l'agente azzurro). Nel secondo caso l'intruso viene inseguito e intercettato da ben due agenti, che in coppia lo bloccano al bordo di Q.

Nella Sezione 2.5.1 del Capitolo 2 si è abbondantemente parlato di come alcuni agenti, nell'eseguire l'effective coverage, possano fermarsi nonostante il task non sia

completo in quanto la TEF attorno a loro risulta nulla. Questo difetto può creare dei problemi nella ricerca dell'intruso. Infatti gli agenti potrebbero arrestarsi prima di aver individuato l'intruso.

Per risolvere questo inconveniente si può aggiungere un vincolo sulla distanza massima fra agenti, come discusso in Sezione 2.5. Tuttavia, senza applicare vincoli che potrebbero risultare poco consoni al task in esame, si può sostituire l'effective coverage con il dynamic coverage task d_{dyn} nell'equazione (5.4). In questo modo gli agenti saranno sempre in movimento. Tuttavia è necessario limitare il valore massimo raggiungibile da d_{dyn} altrimenti per gli agenti sarà più importante eseguire questo task che non inseguire l'intruso.

5.4 Ulteriori sviluppi possibili

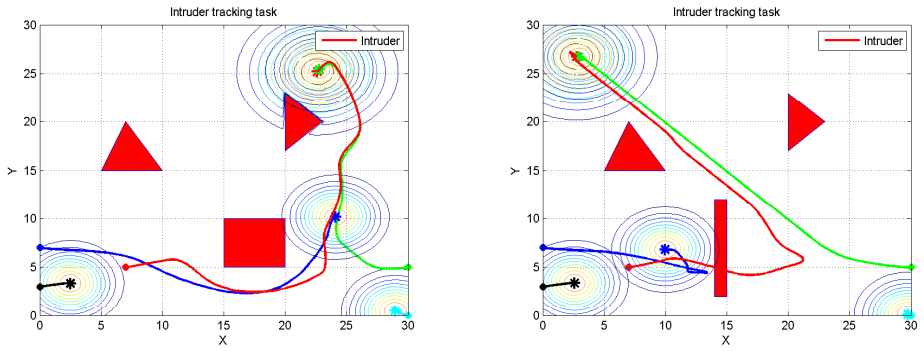
L'analisi fin qui condotta non considera ne vincoli di distanza fra agenti ne la presenza di ostacoli nello spazio Q. L'aggiunta di questi ultimi non comporterebbe peraltro modifiche alle equazioni (5.2) e (5.3) a patto di utilizzare l'opportuna legge di controllo degli agenti che assicuri loro l'obstacle avoidance (equazione (2.5)).

Tuttavia nulla è detto sul comportamento dell'intruso. Non c'è alcuna certezza che esso non possa muoversi attraverso gli ostacoli con la legge di controllo proposta in (5.2).

Sicuramente gli ostacoli svolgono un effetto di repulsione sull'intruso, per lo stesso motivo già illustrato per il bordo di Q. Infatti, poiché la TDF è per ipotesi nulla per $q \in Q_{obs}$, se l'intruso si trovasse vicino ad un ostacolo parte di d_{intr} andrebbe a zero e quindi il valore del controllo u_r diminuirebbe, facendo di conseguenza decrescere J. Siccome però l'intruso vuole massimizzare il funzionale di costo, il controllo lo porterà a stare, per quanto possibile lontano dagli ostacoli.

Si ribadisce tuttavia che non è stata valutata ne una misura esatta di quanto debba essere grande un ostacolo per essere evitato ne vi è nessuna dimostrazione che assicuri l'obstacle avoidance da parte dell'intruso con la legge (5.2). Studi aggiuntivi potrebbero condurre ad una nuova forma per il controllo u_r tale da garantire obstacle avoidance. Si potrebbe ad esempio rimodellare l'idea utilizzata per gli agenti, oppure più semplicemente si potrebbe aggiungere un controllore esterno al Framework predisposto solo ad evitare gli ostacoli.

La differenza con l'effetto provocato dal bordo di Q sta nel fatto che, se l'intruso è posto vicino ad un ostacolo, è possibile che parte della DF superi lo spessore dell'ostacolo stesso. Questa parte che ricompare torna ad essere considerata nel calcolo dell'integrale e quindi dà contributo utile al controllo u_r. Ciò non accade col bordo.



(a) *L'intruso evita l'ostacolo.* **(b)** *L'intruso collide contro l'ostacolo.*

Figura 5.7: Due esempi con ostacoli di grandezza differente: se sufficientemente grandi, gli ostacoli vengono evitati.

Riassumendo, se l'intruso si trova vicino ad un ostacolo ne viene respinto. Tuttavia se è presente un agente, che ha anch'esso un effetto repulsivo, non è detto che l'ostacolo non venga attraversato, a meno che questo non sia sufficientemente grande (deve essere comparabile con la DF). Un esempio di questi due casi si può trovare in Figura 5.7.

Un altro problema aperto di cui si è già discusso all'inizio della Sezione 5.3 è quello dell'identificazione dell'intruso, che si avvicina molto al concetto di cattura in uno scenario pursuit-evader. Il problema risiede in come si possa modellare matematicamente questa astrazione.

Un'idea potrebbe essere quella di definire degli indici dinamici che quantifichino il grado con cui un agente svolge il suo compito. Quando questi superano una soglia predeterminata l'intruso si potrebbe dichiarare identificato. Il primo indice che si può considerare è proprio il valore del funzionale di costo. Se J scende sotto una certa valore allora l'intruso si può considerare catturato.

Un secondo indice, che si basa sulle funzioni descrittive dell'agente k-esimo e dell'intruso, potrebbe essere il seguente:

$$I_k = \int_Q d_k d_{intr} \, \mathrm{d}q \geq 0 \tag{5.5}$$

I_k misura, in modo proporzionale, quanto le DF di agente e intruso sono intersecate, ossia quanto bene l'agente "vede" l'intruso. Nel caso in cui le due funzioni siano uguali, $d_{intr} = d_k$, il massimo valore dell'indice (5.5) si otterrà per $p_k = r$, e $I_k = \int_Q d_{intr}^2$. Per le altre DF non è così banale decidere quando I_k è massimo.

Perciò un buon metodo per decidere la soglia T potrebbe essere:

$$T = \alpha \int_Q d_{intr}^2 \, \mathrm{d}q \qquad \text{con } 0 < \alpha \leq 1 \tag{5.6}$$

L'intruso si può considerare individuato o catturato quando:

$$\sum_k I_k \geq T \tag{5.7}$$

Un'ultima analisi con sviluppi interessanti è quella che prevede più intrusi.

Ovviamente tutte le affermazioni precedenti restano valide: gli agenti tenderebbero a muoversi verso l'intruso più vicino. Tuttavia si potrebbero assegnare dinamicamente gli intrusi ai vari agenti utilizzando proprio gli indici I_k sopra definiti.

Capitolo 6

Agenti con dinamica a doppio integratore

Il Framework delle Funzioni Descrittive, introdotto nei capitoli precedenti, modella gli agenti dello sciame con dinamiche a singolo integratore. Ciò significa che il valore del controllo u rappresenta di fatto la velocità dell'agente. Ovviamente un veicolo o un robot mobile non avrà mai tale dinamica perché affetto da vincoli olonomi, da ritardi e da non linearità.

Ciononostante il singolo integratore è vastamente utilizzato in letteratura. È proprio grazie alla sua semplicità, sia matematica che concettuale, che questa dinamica ha permesso di ottenere importanti risultati in molti campi di applicazione. Senza tale dinamica infatti il DF Framework non potrebbe sfruttare un controllo a discesa del gradiente.

In letteratura alcuni degli algoritmi di Path Planning ([LaValle, 2006]) sfruttano il singolo integratore. Ma soprattutto tale dinamica la si può trovare nel controllo distribuito (si vedano [Lynch, 1996] e [Bullo, Cortés et al., 2009]) e nella teoria sui grafi applicata alle reti di agenti ([Mesbahi e Egerstedt, 2010]). Per fare un esempio si pensi al protocollo del consenso, sia in forma classica che dinamica: gli agenti, che siano effettivamente mobili oppure una rete di sensori distribuiti, sono tutti singoli integratori.

È bene tuttavia considerare che queste teorie, come anche il DF Framework, agiscono ad alto livello. Ciò significa che il controllo non si applica direttamente al veicolo, ma ad una catena di controllo già chiusa e collaudata che cerca il più possibile di movimentare l'oggetto come se avesse una dinamica proprio a singolo integratore.

Ovviamente non sempre si riesce a fare ciò (si pensi a sistemi vincolati), tuttavia in questi casi il formalismo sviluppato sul singolo integratore può fare da base ad

uno sviluppo specifico per il problema in esame.

Principalmente, oltre al singolo integratore, ci sono altre due dinamiche che a volte vengono considerate in letteratura: il doppio integratore e il veicolo di Dubin. Il primo è molto simile al singolo integratore e il controllo fornito a tale sistema rappresenta in sostanza un'accelerazione. Benché sia ancora un modello che semplifica molto la realtà, il secondo ordine riesce a rappresentarla meglio, in quanto include il concetto di inerzia.

Il seconda dinamica invece rappresenta il modello cinematico di un veicolo a due ruote con un raggio di curvatura limitato.

6.1 Dinamica a doppio integratore

Nel Capitolo 1, Sezione 1.3.4, è stato dimostrato che gli agenti del DF FRamework si muovono in maniera tale da condurre il funzionale di costo ad un minimo (locale). Per ottenere tale risultato il funzionale J è stato considerato come una candidata di Lyapunov ottenendo quindi:

$$\dot{J} = \frac{\partial J}{\partial p} \cdot \dot{p} = \frac{\partial J}{\partial p} \cdot u \tag{6.1}$$

dove l'ultimo passaggio si ottiene proprio grazie al fatto che la dinamica dell'agente è un singolo integratore.

Per rendere $\dot{J} \leq 0$ si è applicato quindi un controllo alla Lyapunov, che nel caso particolare risulta a discesa del gradiente:

$$u = -\beta \frac{\partial J}{\partial p} \Longrightarrow \dot{J} = -\beta \Big(\frac{\partial J}{\partial p} \Big)^2 \leq 0$$

Quindi, utilizzando il *Principio di Invarianza* di LaSalle, sono state formulate le opportune conclusioni sulla stabilità e la convergenza del sistema.

Ci si chiede ora se si possa ripetere tale ragionamento con una dinamica del secondo ordine, ossia con:

$$\ddot{p} = u. \tag{6.2}$$

Con questo modello non è più possibile applicare l'ultimo passaggio dell'equazione (6.1), servirebbe infatti la presenza di $\ddot{p}$ e non di $\dot{p}$ all'interno dell'espressione di $\dot{J}$. Perciò, se si vuole continuare ad applicare un controllo alla Lyapunov senza passare per altre tecniche (ad esempio il *Backstepping*), è necessario cambiare la candidata di Lyapunov.

In [Sharifi et al., 2015] si affronta un problema simile a quello appena posto. Nell'articolo, gli autori minimizzano un funzionale di costo utilizzando agenti eterogenei con dinamica a doppio integratore.

Riadattando allora il processo sviluppato in [Sharifi et al., 2015], si sceglie la seguente candidata di Lyapunov:

$$V = J + \frac{1}{2}\sum_{i=1}^{N_a} \|\dot{p}_i\|^2 \geq 0 \tag{6.3}$$

dove $J \geq 0$ è il funzionale utilizzato dal DF Framework. La derivata temporale della candidata di Lyapunov risulta essere:

$$\dot{V} = J_t + \sum_{i=1}^{N_a} \frac{\partial J}{\partial p_i} \cdot \dot{p}_i + \sum_{i=1}^{N_a} \ddot{p}_i \cdot \dot{p}_i = J_t + \sum_{i=1}^{N_a} \Big(\frac{\partial J}{\partial p_i} + \ddot{p}_i\Big) \cdot \dot{p}_i. \tag{6.4}$$

Come è noto dai Capitoli 1 e 2 $J_t = \frac{\partial J}{\partial t}$ risulta nullo o negativo e perciò nel seguito verrà omesso.

A differenza della veloce dimostrazione condotta ad inizio sezione, dove lo stato di tutti gli agenti era racchiuso in $p = [p_1^T, p_2^T, ..., p_{N_a}^T]^T$, in questo caso sono stati esplicitati gli stati di tutti gli agenti tramite pedice, al fine di rendere i passaggi algebrici più chiari possibile. È da sottolineare inoltre come lo stato dell'agente consista sia di posizione che di orientazione: $p_i = [p_{xi}, p_{yi}, \theta_i]^T$.

Nell'equazione (6.4) compare adesso il termine $\ddot{p}_i$ che risulta essere l'ingresso del sistema del secondo ordine.

Proposizione 2. *Utilizzando agenti con dinamica del secondo ordine (equazione (6.2)) e la legge di controllo*

$$\ddot{p}_i = u_i = -\frac{\partial J}{\partial p_i} - \gamma \dot{p}_i, \quad \gamma > 0, \quad \forall i = 1, .., N_a \tag{6.5}$$

allora le traiettorie generate portano il funzionale J in un minimo (locale).

Dimostrazione. Data la legge (6.5), allora la derivata della candidata di Lyapunov in (6.4) diventa:

$$\dot{V} = -\gamma \sum_{i=1}^{N_a} \|\dot{p}_i\|^2 \leq 0 \tag{6.6}$$

che risulta chiaramente semidefinita negativa, è quindi necessario studiare quando diventa nulla. Affinché $\dot{V} \equiv 0$ allora dovrà essere che:

$$\dot{p}_i \equiv 0 \Rightarrow \ddot{p}_i \equiv 0 \Rightarrow u_i \equiv 0 \Rightarrow \frac{\partial J}{\partial p_i} = 0$$

Ciò significa che l'insieme invariante massimo è costituito dai punti per cui $\frac{\partial J}{\partial p_i} = 0$. Perciò, richiamando il *Principio di Invarianza* di LaSalle si può concludere che il sistema si porta in un punto stazionario di J. Esso risulterà essere

□

Anche nel caso in cui il sistema abbia stato iniziale proprio su un punto di massimo o di sella, è sufficiente aggiungere un rumore piccolo a piacere per permettere al sistema di evolvere correttamente.

La dimostrazione appena svolta può essere ripetuta senza alcuna modifica anche nel caso in cui siano presenti gli ostacoli o la Funzione Descrittiva di Connettività. Quello che cambia è la forma esplicita della legge di controllo (6.5), in quanto la derivata parziale di J conterrà i termini in più già analizzati in Sezione 2.2 e 4.3.

6.2 Analisi della legge di controllo

La Proposizione 2 dimostra quindi come il sistema evolva minimizzando J, proprio come accadeva con una dinamica a singolo integratore.

Inoltre quando la velocità degli agenti risulta nulla (non identicamente nulla, ma solo per un istante), si ha:

$$u_i = \ddot{p}_i = -\frac{\partial J}{\partial p_i}$$

ciò significa che l'accelerazione è diretta esattamente lungo la direzione di discesa del gradiente. Il caso appena proposto si verifica sempre ad inizio scenario, momento in cui gli agenti sono sicuramente fermi.

Un altro caso interessante si ha quando $u_i = 0$ (anche in questo caso l'uguaglianza non è da intendersi $\forall t$). Infatti si ottiene:

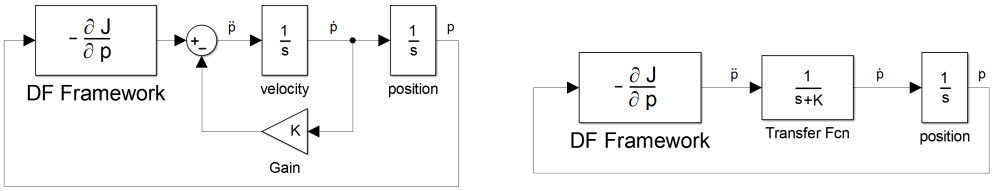
$$\dot{p}_i = -\gamma^{-1}\frac{\partial J}{\partial p_i}$$

che risulta essere proprio la legge di controllo nel caso di agenti modellati come singolo integratore. In pratica l'accelerazione dell'agente può risultare nulla o perché si trova su un minimo locale e quindi poi resterà fermo, oppure perché la direzione che sta seguendo è esattamente quella di massima discesa e non c'è bisogno di modificarla.

In generale il controllo (6.5) imprime all'agente un'accelerazione diretta verso la direzione di massima discesa del funzionale di costo J. A questo si somma il termine di "decelerazione" $-\gamma\dot{p}_i$. Grazie ad esso l'agente non rischia di aumentare la velocità all'infinito e quindi di diventare un sistema instabile.

In Figura 6.1a è stato riportato lo schema a blocchi del "sistema" sciame. Il primo blocco *DF Framework* calcola il gradiente cambiato di segno a partire dalle posizioni degli agenti e dalla TDF (questo ingresso non è rappresentato per semplificare lo schema). A questo termine si aggiunge lo smorzamento della velocità (in figura il termine γ è rappresentato dal guadagno k). Infine si ha il doppio integratore.

In Figura 6.1b è rappresentato lo stesso schema a blocchi dopo aver chiuso l'anello di controllo sul primo integratore. Attraverso questo schema si ottiene una nuova interpretazione della legge (6.5).



(a) **(b)**

Figura 6.1: Schema a blocchi, esplicito e implicito, della legge di controllo ottenuta in (6.5).

Infatti il sistema può adesso rappresentare la dinamica di un agente modellato come singolo integratore, in quanto è rimasto solo un termine $\frac{1}{s}$ nello schema. A differenza del controllo a discesa del gradiente utilizzato nel DF Framework fino a questo momento, l'agente adesso vede come ingresso il gradiente filtrato dalla funzione di trasferimento. Questa ha l'effetto di sfasare e attenuare (con guadagno statico γ^{-1}) il gradiente.

Ciò che si è ottenuto dalla legge di controllo (6.5) non è più un metodo del gradiente ma un metodo *quasi-Newton* (si veda [Bertsekas, 1999]). Anch'essa è una tecnica di ottimizzazione e consiste nello scegliere, come direzione di discesa, una combinazione lineare dei gradienti al passo precedente.

Questa affermazione risulta ancor più evidente se si considera il sistema discretizzato (in questo caso con Eulero in avanti):

$$\begin{aligned} p[n+1] &= p[n] + T\dot{p}[n] \\ \dot{p}[n+1] &= -\delta\dot{p}[n] - TJ_p[n] \end{aligned} \tag{6.7}$$

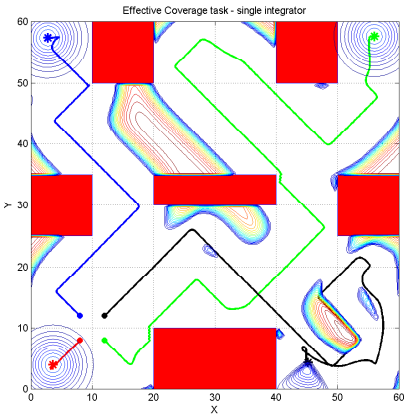
con $-\delta = 1 - \gamma T$ in modulo minore di 1 altrimenti il sistema risulterebbe instabile, T tempo di campionamento del sistema e $J_p = \frac{\partial J}{\partial p}$

Considerando per semplicità $T = 1$, i primi tre passi del sistema (6.7) saranno:

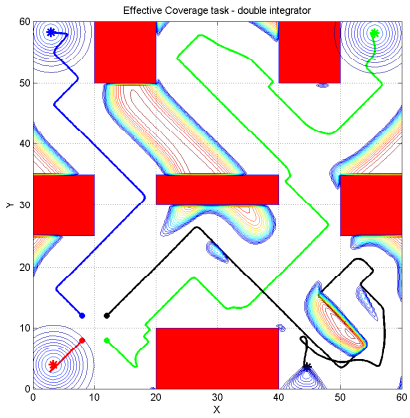
$$\begin{aligned} p[1] &= 0 \\ p[2] &= -J_p[0] \\ p[3] &= (1-\delta)J_p[0] - J_p[1] \end{aligned}$$

Dall'ultima espressione risulta chiaro come le varie posizioni in cui si porta l'agente siano una combinazione lineare dei gradienti ai passi precedenti.

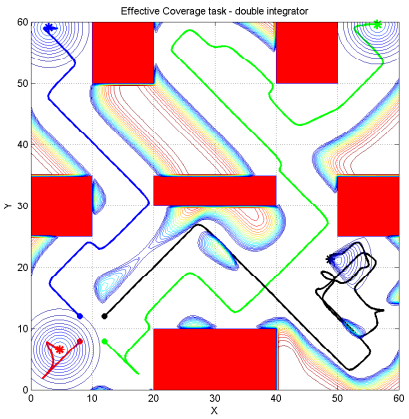
Infine si vuole porre l'attenzione su come il funzionale J sia del tutto generico e possa includere gli ostacoli, la Funzione Descrittiva di Connettività e anche i vincoli di distanza inter-agente.



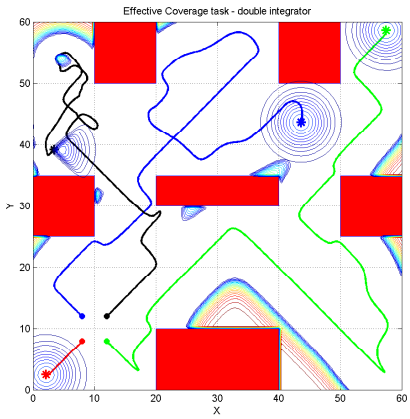
(a) *Singolo integratore*



(b) *Doppio integratore con* $\delta = -0.1$



(c) *Doppio integratore con* $\delta = -0.5$



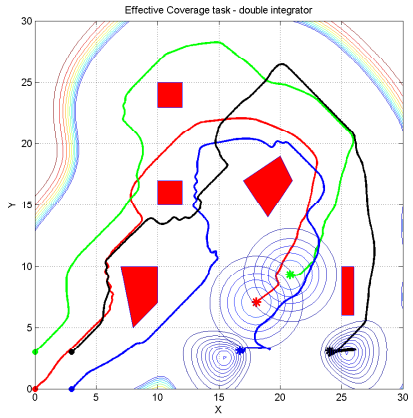
(d) *Doppio integratore con* $\delta = -0.8$

Figura 6.2: Effective Coverage Task eseguito con dinamica a singolo integratore e dinamica a doppio integratore con vari guadagni

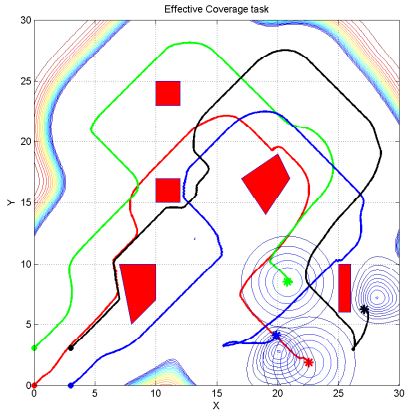
6.3 Conclusioni ed esempi

In questo capitolo si è dimostrato come modificare la legge di controllo del DF Framework affinché gli agenti possano essere descritti con una dinamica a doppio integratore. Tale dinamica, seppur semplice, si accosta meglio ai processi reali rispetto al singolo integratore fin qui utilizzato. Inoltre è stato mostrato come la legge ottenuta possa essere interpretata come un metodo di ottimizzazione *quasi-Newton*.

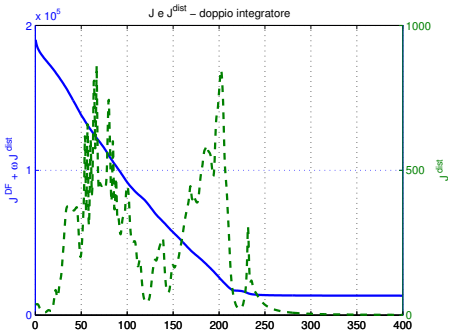
In Figura 6.2 si possono osservare quattro simulazioni che considerano gli agenti nelle stesse condizioni iniziali a cui è richiesto di eseguire un task di effective coverage. La Figura 6.2a mostra il percorso effettuato considerando la dinamica a singolo integratore. Nella Figura 6.2b si utilizza invece una dinamica a doppio integratore. Le due traiettorie sono molto simili. Il motivo è da ricercarsi nel valore del guadagno



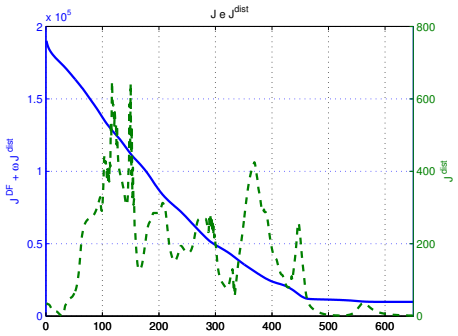
(a) *Percorso con* $\delta = -0.5$



(b) *Percorso singolo integratore*



(c) *Funzionali di costo con* $\delta = -0.5$



(d) *Funzionali di costo singolo integratore*

Figura 6.3: Effective Coverage Task eseguito con dinamica a doppio integratore (a sinistra) e a singolo integratore (a destra) con attivi i vincoli sulle distanze.

δ del sistema discreto (6.7). Nel caso in esame $\delta = -0.1$. Questo significa che il valore di $\dot{p}[n]$ influisce molto poco sul valore di $\dot{p}[n+1]$. In pratica $\dot{p}[n+1] \approx \frac{\partial J}{\partial p}$ che risulta essere una dinamica a singolo integratore.

Lo stesso ragionamento può essere esposto considerando il sistema tempo continuo di Figura 6.1b. In questo caso è come se γ (k in figura) avesse un valore molto elevato. In questo la funzione d'anello $\frac{1}{s(s+\gamma)}$ è approssimabile al singolo integratore, in quanto il polo in $-\gamma$ è ad alta frequenza.

Aumentando il valore di δ, o viceversa abbassando quello di γ, si inizia a vedere una certa differenza nelle traiettorie. In particolare, in Figura 6.2d, il percorso dell'agente nero risulta completamente differente dai casi precedenti. Anche gli altri agenti ottengono una traiettoria leggermente modificata.

In Figura 6.3 sono messi a confronto sempre due task di effective coverage ma in questo caso sono attivi i vincoli sulle distanze inter-agente sviluppati nel Capitolo 3. Lo scenario è lo stesso presentato in Sezione 3.4.

Benché le traiettorie generate siano molto simili, e anche il valore finale del funzionale $J = J^{DF} + \omega J^{dist}$ risulti pressapoco uguale, quello che cambia è il tempo impiegato dagli agenti per eseguire la simulazione. Nell'esempio di Figura 6.3c il funzionale smette di decrescere sostanzialmente a 300 passi di campionamento. Nell'altro caso si impiega circa il doppio del tempo. Ciò accade perché in generale i metodi *quasi-Newton* sono più veloci del metodo del gradiente. Bisogna stare tuttavia attenti a scegliere con cura il valore γ.

6.3.1 Note sulle simulazioni

Le simulazioni, presentate in questo e negli altri capitoli, sono state sviluppate in Matlab®/Simulink. Quindi, nonostante tutta la teoria sviluppata sia tempo continua, le simulazioni sono state svolte considerando processi discreti. Anche se Simulink fornisce la possibilità di operare in tempo continuo, in realtà utilizza soluzioni discrete con passo di campionamento variabile. Si è preferito allora implementare direttamente tutte le leggi nel discreto, in modo da avere un controllo più fine sulle simulazioni.

Il tempo di campionamento scelto è perciò di fondamentale importanza in quanto limita l'ampiezza del gradiente e delle azioni di controllo. Infatti, come è noto dalla letteratura, se il passo del gradiente risulta troppo elevato, non solo si possono perdere soluzioni (i minimi del funzionale di costo), ma si può anche entrare in oscillazione.

Per limitare al massimo questi fenomeni è stato scelto di aggiungere una saturazione al valore del controllo. Si vuole infine porre l'accento sul fatto che questa soluzione non modifica il comportamento dello sciame. Infatti si sta semplicemente imponendo un modulo massimo al gradiente calcolato analiticamente, non si stanno modificando quelle che sono le informazioni principali ad esso associate associate, ossia la direzione ed il verso.

Anche la scelta del tempo di campionamento può variare da caso a caso. Infatti se si sta eseguendo una simulazione contente ostacoli e vincoli sulle distanze implementate secondo le leggi dei Capitoli 3 e 4 è importante mantenere il passo di avanzamento il più piccolo possibile. Per fare un esempio, si potrebbe passare dal punto $p[n]$ lontano dagli ostacoli, al punto $p[n+1]$ che invece ne risulta molto vicino. Conseguentemente il controllo $u[n+1]$ diverrà improvvisamente molto elevato. Potrebbe anche succedere che l'agente si ritrovi al tempo $n+1$ dentro l'ostacolo oppure oltre la distanza massima/minima imposta.

Nel caso di dinamica del secondo ordine il passo di campionamento varia a seconda di δ. Se $-\delta$ è alto, vicino a 1, è necessario utilizzare un tempo di campionamento più piccolo. Questo perché il sistema, all'aumentare di δ, aumenta la sua inerzia, e risulta quindi meno pronto a rispondere ad eventuali forze repulsive o di richiamo (casi con ostacoli e vincoli sulle distanze). Perciò, un passo di campionamento piccolo assicura che il sistema abbia tutto il tempo per rallentare e cambiare direzione. Per

fare un esempio, nella simulazione di Figura 6.2d, dove $\delta = 0.8$, è stato necessario utilizzare un tempo di campionamento cinque volte inferire a quello degli altri esempi di Figura 6.2.

Anche il ragionamento esposto alla fine della sezione precedente riguardo la velocità di convergenza deve essere soppesato al tempo di campionamento utilizzato. Non è corretto definire un algoritmo o un processo più veloce di un altro, se questo usa passi campionamento differenti dalle altre tecniche. Nell'esempio di Figura 6.3, le due simulazioni eseguono con il medesimo passo, così che il confronto risulti corretto, anche se, ad osservare il percorso di Figura 6.3a, sarebbe stato meglio abbassare il tempo di campionamento così da rendere il tracciato più dolce.

Capitolo 7

Validazione sperimentale

Nei capitoli precedenti è stato mostrato come il Framework delle Funzioni Descrittive permetta di controllare sciami di agenti, riuscendo a garantire obstacle avoidance e vincoli sulle distanze massime fra gli agenti stessi. Il lavoro svolto tuttavia è puramente teorico e anche gli esempi mostrati nelle figure sono simulazioni svolte al computer.

Per validare l'apparato teorico sviluppato è stato deciso di implementarla in una applicazione reale. Per fare ciò, è stato necessario sviluppare dei veicoli autonomi che, dotati degli opportuni sensori, potessero realizzare parte dei controlli esaminati nel corso della tesi.

Nello specifico sono state utilizzate due "mini-car" (presentate in Figura 7.1) che svolgono il ruolo di due agenti, il resto dello sciame rimane invece virtuale.

Sfortunatamente il DF Framework richiede un carico computazionale troppo elevato per poter essere eseguito direttamente sul microcontrollore che governa i veicoli. Perciò è stato deciso di eseguire la simulazione su pc e, in tempo reale, inviare alle "mini-car" le posizioni degli agenti come se fossero dei waypoint da raggiungere.

Figura 7.1: Veicoli autonomi sviluppati per la prova sperimentale.

Queste, attraverso una guida LOS Steering raggiungono autonomamente la posizione comandatagli.

È doveroso sottolineare come i waypoint siano generati da tutti gli agenti simulati. Perciò, se si avesse avuto la disponibilità di una terza o quarta "mini-car" , si sarebbe potuto svolgere la prova con più agenti reali senza modificare assolutamente nulla del sistema realizzato.

7.1 Mini-Car

Le "mini-car" sono delle *HSP 1/10 Scale Brontosaurus Electric 4WD Monster Truck*. Dell'equipaggiamento già presente a bordo sono stati riutilizzati soltanto il motore principale, il suo driver e lo sterzo. Il resto dell'elettronica è stato aggiunto e programmato ex-novo.

Le principali caratteristiche fisico-meccaniche dei veicoli sono:

- Alimentazione con batteria da modellismo da $7.2V$;
- Controllo Elettronico della velocità ESC;
- Quattro ruote motrici;
- Larghezza: $310mm$, Lunghezza: $400mm$, Altezza: $185mm$;
- Peso: $3Kg$. Carico massimo: $1Kg$.

Figura 7.2: Mini-car

Per poter controllare correttamente i motori, avere i riferimenti inerziali per la guida LOS, e permettere la comunicazione con il pc sono stati aggiunti i seguenti componenti:

STM32F4 : è il microcontrollore che gestisce tutti i dispositivi a bordo dei veicoli, processa le informazioni dei sensori, comanda i motori e si preoccupa di eseguire correttamente la legge di guida LOS;

Xbee-Pro S1 : è il modulo di comunicazione wireless che permette lo scambio di dati tra macchine e pc;

Ublox LEA-4H / NEO-6 : sono i moduli GPS che forniscono le informazioni di posizione e velocità al microcontrollore. Inoltre forniscono un segnale periodico (noto come PPS) utilizzato per sincronizzare le comunicazioni (di questo si parlerà dettagliatamente più avanti);

HMC5883L : magnetometro a tre assi necessario per conoscere l'orientazione del veicolo nello spazio.

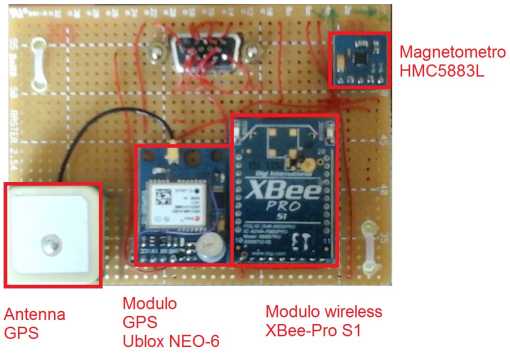


Figura 7.3: Sensori e modulo di comunicazione montati sui veicoli.

L'alimentazione a tutta l'elettronica viene fornita dalle batterie, tuttavia deve essere abbassata la tensione da 7.2 a 5 Volt. A tal fine è presente un convertitore DC/DC **TRACOPOWER TSR 1-2450**. Per avere invece tensioni a 3.3V, necessarie per alimentare il magnetometro, si sfrutta il convertitore presente sulla board del microcontrollore.

7.1.1 Modello cinematico

Il moto dei veicoli utilizzati può essere descritto da un modello cinematico abbastanza semplice che assume le condizioni di sterzata di Ackermann. In Figura 7.4 è rappresentato il modello utilizzato.

Le variabili generalizzate che descrivono il moto del veicolo sono $[x, y, \theta]$ con x e y posizione del centro dell'assale posteriore, mentre θ è l'angolo formato dall'asse del veicolo con le ascisse. Gli ingressi invece risultano essere $[v, \varphi]$, il primo è la velocità dell'assale posteriore, mentre φ è l'angolo di sterzo relativo a θ.

Le equazioni del modello cinematico risultano essere:

$$\begin{cases} \dot{x} = \cos(\theta)v \\ \dot{y} = \sin(\theta)v \\ \dot{\theta} = \frac{1}{L}\tan(\varphi)v \end{cases} \tag{7.1}$$

In realtà i veicoli utilizzati non hanno la possibilità di sterzare di un angolo φ qualsiasi, come invece può sembrare dal modello. L'angolo di sterzata massima

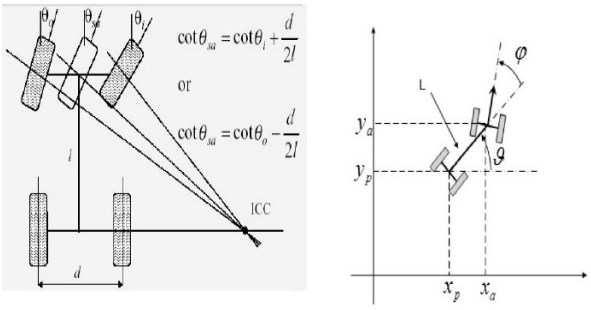


Figura 7.4: Modello cinematico dei veicoli.

risulta:

$$|\varphi| \leq \frac{\pi}{12}$$

che può essere modellato, dal punto di vista sistemistico, come una saturazione.

7.2 Modulo GPS

Il modulo GPS utilizzato sulle macchine è della serie **UBLOX**. Su una macchina è presente il più moderno **NEO-6**, sull'altra il più datato **LEA-4H**.

Entrambi i moduli comunicano attraverso interfaccia seriale *RS232* con 8 bit di payload, 1 bit di stop e nessun bit di parità. Al fine di comunicare il più velocemente possibile il *baud rate* è stato impostato a 155200 bit/sec.

I moduli UBLOX implementano un firmware molto versatile e con parecchie funzionalità. Nell'applicazione sviluppata tali estensioni sono state disabilitate in modo da avere le informazioni sulla navigazione più pulite possibili, senza filtraggi o modifiche nascoste all'utente finale.

Il protocollo di comunicazione settato è il *binario ubx*, formato proprietario della UBLOX. In particolare si è abilitato soltanto il pacchetto *NAV-SOL* contenente una serie di informazioni sulla navigazione. Le seguenti sono quelle realmente utilizzate per l'applicazione:

- Posizione XYZ in centimetri, con sistema di riferimento ECEF;
- Velocità XYZ in cm/s, con sistema di riferimento ECEF;
- Fix del GPS;
- Status Flag, per validare il dato precedente sul Fix.

Nonostante la posizione e la velocità siano date in centimetri e centimetri al secondo l'accuratezza è dell'ordine del metro. Questo valore cambia particolarmente a seconda delle condizioni in cui si trova l'antenna. Infatti se una parte del cielo

Tabella 7.1: Accuratezza e *time to fix* medi dei moduli GPS in condizioni ottimali.

Modulo	Accuratezza Posizione [m]	Accuratezza Velocità [m/s]	Time to Fix [s]
LEA-4H	4	0.3	60
NEO-6	2.5	0.2	20

risulta coperta, non vengono ricevuti i segnali da alcuni satelliti e quindi l'accuratezza decresce (cioè l'errore in metri aumenta).

Invece migliora sensibilmente quando il cielo risulta completamente libero da ostacoli. In Tabella 7.1 sono riportati i valori medi che si ottengono in questo caso e il *time-to-fix* medio, ossia quanto tempo impiega il modulo, dal momento dell'accensione, a fornire la prima stima di posizione valida.

Infine entrambi i moduli GPS possiedono un pin d'uscita detto PPS o *Pulse Per Second*. Il segnale in uscita da questo piedino è un'onda quadra con periodo impostato a $1Hz$. Il suo andamento nel tempo non è generato dal clock interno del modulo GPS ma è allineato con il secondo UTC ricevuto dal sistema satellitare. Risulta perciò estremamente accurato e sincrono su tutti i dispositivi. Di tutto ciò si parlerà nel dettaglio in Sezione 7.7.

7.3 Magnetometro

Il magnetometro **HMC5883L** è un sensore a tre assi che fornisce il vettore 3D del campo magnetico nelle sue vicinanze. La comunicazione con il microcontrollore avviene tramite protocollo I^2C e non via seriale. Esso viene utilizzato per conoscere l'orientamento del veicolo sul piano. Montando infatti l'asse X del sensore allineato con la testa della "mini-car" e l'asse Y verso destra (quindi con asse Z rivolto verso il basso), l'angolo θ presente fra macchina e NORD magnetico è calcolabile come[1]:

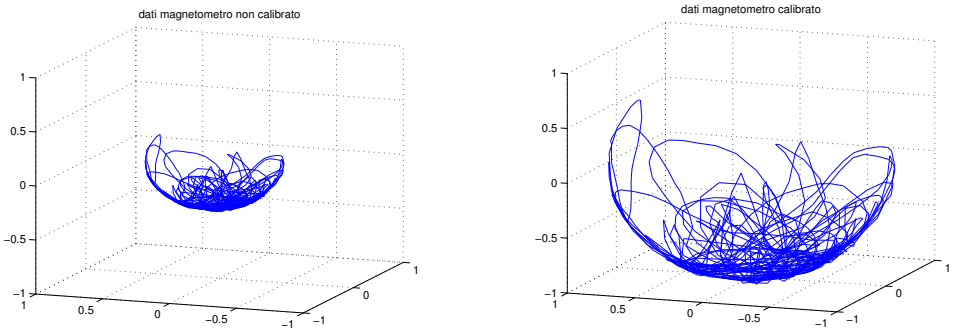
$$\theta = -\arctan(\frac{mag_y}{mag_x}) \tag{7.2}$$

In questo modo per $\theta = 0$ il veicolo punta a NORD, per $\theta = 90°$ punta a EST e così via. La precisione ottenuta, senza alcun filtraggio sui dati ricevuti (ma con opportuna calibrazione), è di $\pm 3°$.

Il magnetometro tuttavia è molto sensibile a materiali ferromagnetici e soprattutto al motore a magneti permanenti che muove la "mini-car" . Per questo motivo è stato montato sul pennone, assieme al modulo GPS e Xbee. Ad una tale distanza il motore non interferisce più con la misura del campo magnetico terrestre.

Infine va ricordato che il magnetometro, più di tanti altri sensori, deve essere calibrato prima di poter essere utilizzato correttamente. La calibrazione è stata

[1]Nell'implementazione Matlab viene utilizzato -atan2(mag_y,mag_x)



(a) Dati non calibrati **(b)** Dati calibrati

Figura 7.5: Dati del magnetometro prima e dopo la calibrazione

eseguita seguendo il metodo proposto in [Merayo et al., 2000]. In pratica si raccolgono i dati dal sensore non calibrato cercando di muoverlo in tutto lo spazio 3D. Tali dati, mostrati in Figura 7.5a, assumeranno la forma di un ellissoide non centrato nell'origine. Quello che invece si vorrebbe ottenere è una sfera di raggio unitario con centro nello 0.

Per ottenere la forma desiderata, si calcolano, attraverso un processo di ottimizzazione ai minimi quadrati, una matrice diagonale di scalatura $U \in \mathbb{R}^{3\times3}$ e un vettore di offset $c \in \mathbb{R}^3$. I dati calibrati correttamente si ottengono dalla seguente trasformazione:

$$mag_{calibrato} = U(mag_{grezzo} - c) \tag{7.3}$$

dove $mag_{calibrato}$ e mag_{grezzo} sono vettori di tre elementi ed indicano rispettivamente i dati calibrati e i dati che arrivano dal sensore.

In Figura 7.5 si può notare come il mezzo ellissoide ottenuto dai dati grezzi diventi, dopo la trasformazione (7.3), più sferico, con raggio quasi unitario e con centro nell'origine.

7.4 Modulo Xbee

Il modulo **XBee-Pro S1** è sostanzialmente un'interfaccia di comunicazione radio, wireless. A livello di microcontrollore quest'unità viene vista semplicemente come una porta seriale *RS232* a cui spedire e da cui ricevere dati. Il *baud rate* in questo caso è stato impostato a 57600 bit/s, al di sotto del massimo utilizzabile. Infatti sono stati riscontrate grosse complicazioni nell'utilizzo di un baud rate più elevato. Tali problemi derivano dall'oscillatore interno al modulo che non riesce a sostenere correttamente una frequenza più elevata. Per maggiori informazioni si veda [Foster, 2011].

Proprio come per il GPS, anche in questo caso il modulo XBee offre molte funzionalità settabili attraverso un programma specifico (X-CTU). Molte di queste permettono di instaurare canali di comunicazione peer-to-peer con altri XBee. Tuttavia, siccome l'applicazione in questione richiede una comunicazione molti a uno e viceversa, tutte queste estensioni sono state disabilitate (in particolare le opzioni *MaxStream* e *ACK*)rendendo di fatto lo XBee un modulo che invia pacchetti di dati nell'etere.

L'unica impostazione mantenuta è quella che prevede di scartare pacchetti destinati ad altri moduli XBee. Infatti, poiché l'informazione inviata viaggia su onde radio, essa è, per forza di cose, broadcast. Perciò è necessario far si che i dati spediti da una "mini-car" arrivino soltanto al modulo XBee collegato con il pc.

Per fare ciò ogni unità montata sui veicoli è contrassegnata da un numero interno progressivo (registro MY) a partire da 1. Il numero 0 è assegnato invece al modulo del pc. Dopodiché settando l'indirizzo del destinatario a 0 si fa in modo che le macchine inviino soltanto alla stazione base.

Viceversa, il pc deve inviare un solo pacchetto a tutti i veicoli, e cioè spedire in broadcast. Perciò in questo caso il destinatario è impostato come 0x0000FFFF.

7.5 Microcontrollore

Il microcontrollore utilizzato è un **STM32F4** della ST Microelectronics ed è risultato potente e molto versatile. Attraverso il micro si inviano e ricevono informazioni alle varie unità descritte in precedenza. Inoltre si comandano i motori delle "mini-car" attraverso la generazione di due segnali PWM.

La programmazione della scheda è stata effettuata attraverso Simulink e in particolare è stato utilizzato il *MAT-Target STM32F4xx V2* e *Keil µVision4* come toolchain di compilazione.

L'utilizzo di Simulink, come sistema si programmazione, permette di costruire dei modelli real-time ben definiti e soprattutto modulari. Questo fatto non è da trascurare. Infatti qualora si volesse cambiare, modificare o eliminare una parte del "sistema veicolo", basterebbe agire sui singoli blocchi Simulink senza modificare una linea di codice.

Tuttavia la sezione del firmware riguardante le interruzioni e l'utilizzo dei moduli interni al microcontrollore, come porte USART, I^2C, Timer e PWM, è stata scritta direttamente in C utilizzando le *Standard Peripheral Library* e integrata successivamente in Simulink.

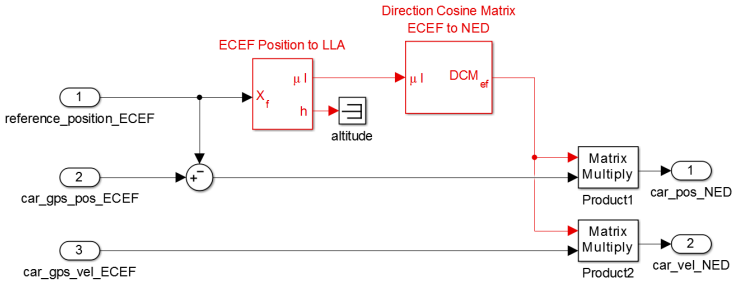


Figura 7.6: Schema a blocchi che realizza la conversione da ECEF a NED.

7.6 Autopilota dei veicoli

L'autopilota dei veicoli è un sistema complesso, comprensivo di un sistema di guida, uno di navigazione e di un controllo di basso livello dei motori.

Per quel che riguarda la parte di navigazione si ricorda che le macchine sono dotate di magnetometro e sensore GPS da cui ricavare orientazione, posizione e velocità del veicolo stesso. Tuttavia i dati del GPS sono in coordinate ECEF, è perciò necessario trasformarle in coordinate NED prima di poterle utilizzare. Per questo motivo i veicoli ricevono, attraverso il sistema di comunicazione, la posizione, in coordinate ECEF, del centro del sistema di riferimento NED. In Figura 7.6 è possibile vedere lo schema a blocchi utilizzato per la conversione da ECEF a NED.

Sui veicoli è implementato un sistema di guida LOS che si preoccupa di condurre le "mini-car" dalla loro posizione attuale verso un waypoint. Come tutte le guide, essa di occupa di generare i riferimenti utilizzati dal sistema di controllo.

La LOS è tecnica particolarmente semplice ma efficace. Essa prevede di variare l'angolo di sterzo finché la direzione di moto del veicolo non risulti puntare dritta verso il waypoint. Congiuntamente genera un riferimento di velocità, proporzionale alla distanza dal waypoint, cosicché il veicolo possa variare la sua orientazione (orientazione e velocità sono legate, si veda l'equazione (7.1)) e raggiungere il target.

A livello matematico, detti $p = [p_x, py]$ e $w = [w_x, w_y]$ la posizione rispettivamente del veicolo e del waypoint rispetto ad un riferimento dato, si calcola[2]:

$$\theta_{ref} = \arctan(\frac{w_y - p_y}{w_x - p_x}) \tag{7.4}$$

$$v_{ref} = f(\|p - w\|) \tag{7.5}$$

dove $f(\cdot)$ è una funzione non decrescente della distanza tra waypoint e veicolo.

Nel caso in esame, vista l'incertezza sulla posizione del veicolo pari a circa 2 metri, dovuta all'accuratezza del sensore GPS, è stato deciso di considerare il waypoint come raggiunto quando esso si trova ad un raggio di due metri dalla "mini-car" .

[2]Nell'implementazione Matlab viene utilizzato -atan2($w_y - p_y$,$w_x - p_x$)

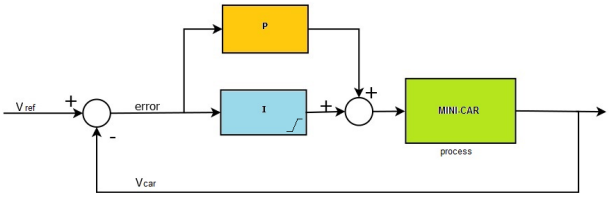


Figura 7.7: Modello del regolatore PI sulla velocità.

Tabella 7.2: Guadagni dei regolatori PI e soglie anti-windup.

	K_p	K_i	anti-windup [min/max]
PI motore	0.3	0.15	0/1
PI sterzo	0.6	0.075	$-0.273/0.273$

Perciò la velocità viene portata a 0 quando macchina e waypoint distano meno di 2 metri. Ovviamente a causa dell'inerzia, il veicolo non raggiungerà immediatamente velocità nulla, ma continuerà a muoversi per un breve tratto, col risultato di finire la maggior parte delle volte molto vicino al waypoint desiderato.

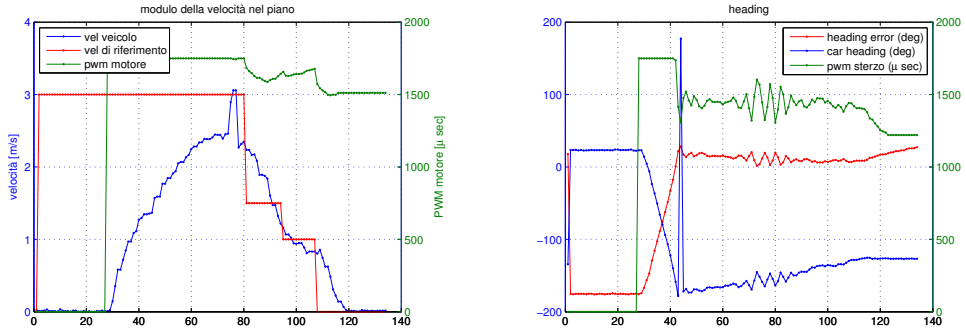
La forma della funzione $f(\cdot)$ dell'equazione 7.5 risulta:

$$v_{ref} = f(d) = \begin{cases} 3m/s, & \text{if } d \geq 10m \\ 1.5m/s, & \text{if } 5m < d < 10m \\ 1m/s, & \text{if } 2m < d \leq 5m \end{cases} \tag{7.6}$$

Intanto che la legge di guida genera i riferimenti v_{ref} e θ_{ref}, il controllo di basso livello si occupa di portare il sistema "mini-car" allo stato desiderato. Per fare ciò sono stati implementati due regolatori PI tempo discreto. Il primo, agendo sul motore cerca di portare la velocità, ricavata dal sensore GPS, a quella desiderata. Il secondo invece si occupa dell'orientazione. Esso ruota lo sterzo finché $\theta = \theta_{ref}$. La variabile θ è ottenuta dalla lettura dei dati del magnetometro secondo l'equazione (7.2).

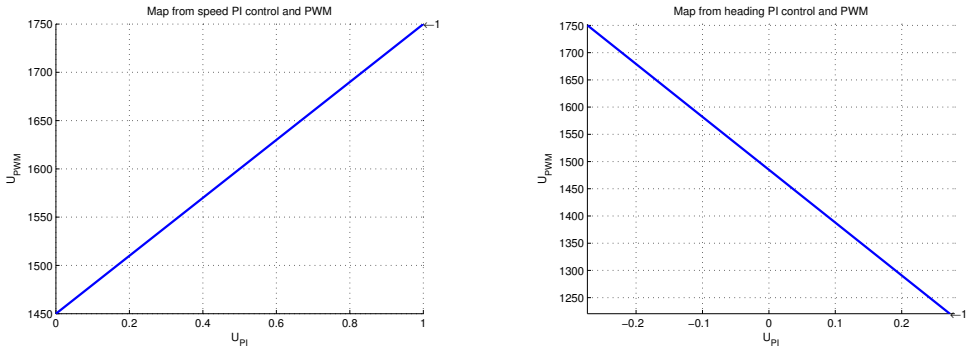
Ovviamente, per evitare sovraelongazioni eccessive e effetti di *windup*, l'integratore discreto è stato dotato di una saturazione sullo stato interno. Questo significa che, superata la soglia limite, superiore o inferiore, il blocco smette di sommare e mantiene costante l'uscita fino a che essa non ritorna all'interno della soglia.

In Tabella 7.2 sono riportati i guadagni dei PI e le soglie della saturazione *anti-windup*. Questi sono stati tarati sperimentalmente osservando il moto della macchina diretta verso un waypoint prefissato. In Figura 7.8 sono riportati gli andamenti di velocità e sterzo ottenuti dai sensori rispetto ai valori di riferimento e alle uscite PWM comandate ai motori durante la fase di taratura.



(a) **(b)**

Figura 7.8: Andamento di velocità e sterzo rispetto ai riferimenti e alle uscite PWM durante la fase di taratura del PI.



(a) *Mappa motore* **(b)** *Mappa sterzo*

Figura 7.9: Mappe lineari per riportare l'uscita dei regolatori PI in μs PWM.

L'uscita del regolatore PI, sia dello sterzo che della velocità, non è però adatta ad essere inviata ai motori. Infatti questi vengono comandati tramite PWM definito dai microsecondi in cui l'onda quadra deve restare positiva. È stato perciò necessario tradurre il valore di controllo generato dai PI in un valore in μs. Ciò è stato realizzato attraverso le due mappe lineari riportate in Figura 7.9.

Al valore in uscita dalle mappe è infine applicata una saturazione. Ciò si è reso necessario per evitare che, per qualsiasi motivo, arrivassero ai motori valori PWM fuori scala o eccessivi.

In Figura 7.10 è illustrato lo schema d'insieme del funzionamento delle "mini-car" che può essere riassunto secondo i seguenti step:

1. Dai sensori e dalla posizione del waypoint il sistema di guida LOS calcola i riferimenti;

2. I regolatori PI generano il valore di controllo per inseguire i riferimenti;

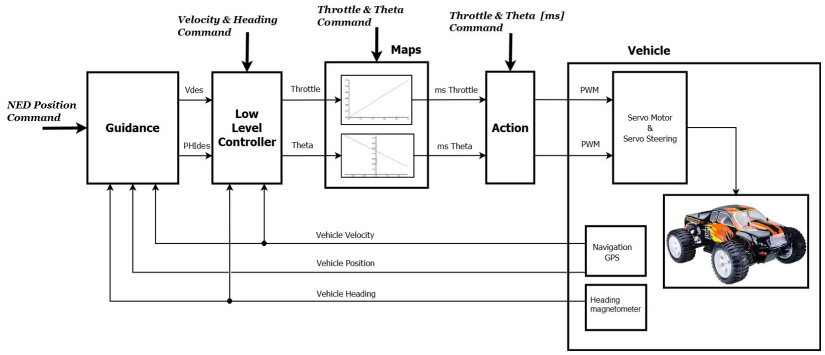


Figura 7.10: Schema completo del funzionamento dei veicoli autonomi.

3. Le mappe traducono il comando dei PI in un valore in μs da inviare al Timer del microcontrollore che genera a sua volta i segnali PWM di pilotaggio dei motori di trazione e sterzo.

7.7 Sistema di comunicazione

Il sistema di comunicazione è uno dei punti più delicati di tutta la struttura hardware/software sviluppata. Senza di esso non si potrebbero svolgere né le prove sperimentali né i test per la taratura dei guadagni dei PI. Inoltre verificare il corretto funzionamento di tutta l'infrastruttura e le sue performance non è semplice in quanto eseguire un debug sul programma caricato sul microcontrollore risulta spesso poco pratico.

Per consentire ai veicoli e al pc di comunicare senza che avvengano collisioni nell'etere, è stato implementato un protocollo di tipo *Time Division Multiple Access* o TDMA. Questa tecnica suddivide il tempo in *time frame* (o intervalli di tempo) e poi divide ciascun intervallo in N slot. A ogni elemento dell'infrastruttura di comunicazione (macchine e pc) è assegnato uno slot ben determinato. Ogni volta che tale elemento deve inviare informazioni aspetta che arrivi il proprio slot all'interno del frame e quindi trasmette.

Nell'applicazione sviluppata il pc comunica nello slot 0. Alla prima "mini-car" invece è assegnato lo slot numero 1 e così via. Il sistema è stato sviluppato per 10 elementi, in pratica come se ci fossero nove agenti e la stazione base, in quanto si ha intenzione di ampliare il testbed in futuro.

Ogni time frame è lungo $200ms$, di conseguenza ogni slot temporale ne dura 20. Considerando che la comunicazione tra microcontrollore e XBee avviene a $57600bit/s$ in ogni slot temporale possono essere inviati pacchetti non più lunghi di 144 byte, che corrispondono a 36 float, numero più che sufficiente per scambiare correttamente tutte le informazioni.

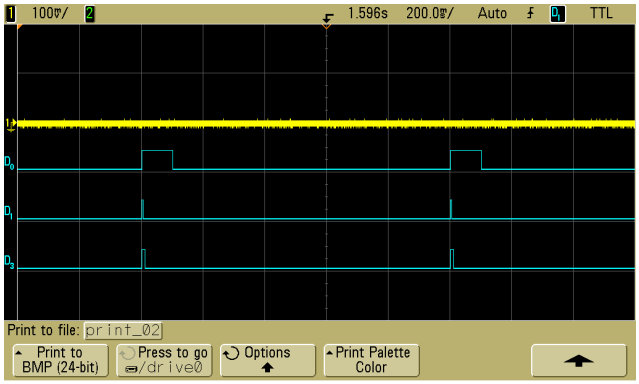


Figura 7.11: Segnale PPS di tre moduli GPS di marche differenti.

Il protocollo TDMA funziona correttamente anche eliminando l'ipotesi che vede il modulo XBee trasmettere i dati nel momento in cui li riceve. Infatti, anche se lo XBee aspettasse di aver ricevuto dal microcontrollore un certo numero di byte prima di inviarli (cosa che avviene realmente), questo si avrebbe per tutti i moduli dell'infrastruttura di comunicazione traslando di fatto tutti gli slot.

Tuttavia, per implementare correttamente il protocollo TDMA è necessario un clock comune a tutte le parti in gioco o un segnale che scandisca l'inizio di un time frame. Nell'applicazione sviluppata è stato utilizzato il segnale PPS proveniente dai moduli GPS, seguendo l'approccio svolto in [Della Santina e Stefano, 2011].

Il PPS è un onda quadra con un periodo esattamente di un secondo. Tale segnale viene generato dai GPS sulla base del secondo UTC che ricevono dal sistema satellitare e non da un oscillatore interno. Perciò il PPS risulta un segnale stabile, preciso e soprattutto comune a tutte le unità GPS.

Per testare l'efficacia di questa di questa soluzione sono stati effettuati una serie di esperimenti con moduli GPS di marche differenti. In Figura 7.11 si possono osservare le onde quadre generate da tre dispositivi: un Ublox lea-4h, un Venus638 e un MTK MT3339. Anche se l'ampiezza dell'onda quadra risulta differente, si nota come il periodo sia estremamente preciso, in fase e indipendente dal tipo di ricevitore.

Attraverso questo segnale a $1Hz$ il microcontrollore riesce a individuare l'istante di inizio del time frame. Infatti è stato implementato un contatore che aumenta di una unità ad ogni millisecondo (dato dall'oscillatore interno della scheda). Il segnale PPS si limita a resettare questo modulo. Detto t il valore in uscita del contatore, l'inizio di un time slot con ID I si avrà quando $mod(t, 200) - 20I = 0$. In questo modo i veicoli sono in grado di riconoscere con precisione l'inizio del proprio slot temporale.

Tuttavia, mentre ai microcontrollori delle "mini-car" può essere collegato il segnale PPS, questo non può avvenire per il pc. Per risolvere questo problema è stata sviluppata una base station sempre attraverso un microcontrollore **STM32F4** che

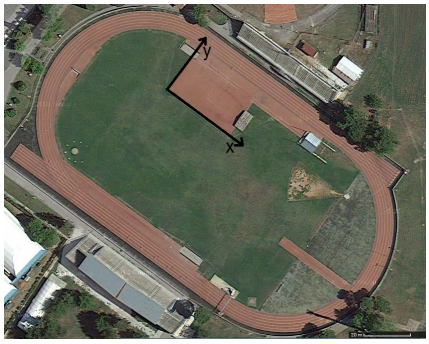


Figura 7.12: Vista in pianta dell'area test.

svolge semplicemente il ruolo di ponte fra pc e modulo di comunicazione XBee. Al microcontrollore è però connesso il modulo GPS e quindi le comunicazioni possono avvenire, anche in questo caso, in perfetta sincronica.

7.8 Test

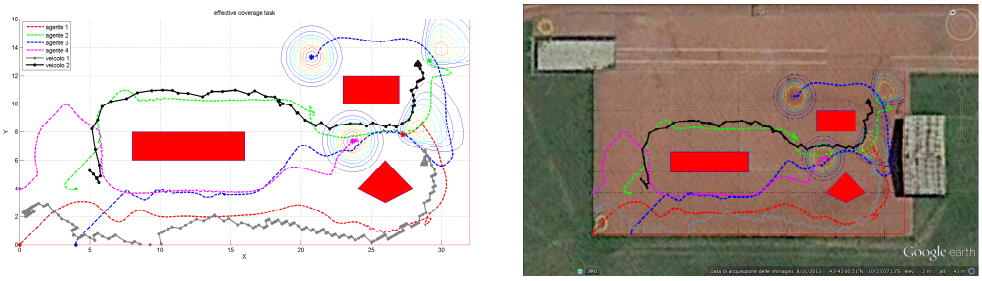
I test per la validazione sperimentale del corretto funzionamento del DF Framework sono stati svolti al campo sportivo CONI di Pisa. Per la precisione nell'area dedicata alla pedana del salto in alto, Figura 7.12. Lo spazio operativo risulta un rettangolo di 32×16 metri ruotato rispetto al NORD di $126°$. Sfortunatamente quest'area risulta un poco ristretta rispetto alle incertezze che affliggono i dati di navigazione provenienti dal GPS. Tuttavia sono stati ottenuti buoni risultati.

Durante le varie giornate di test sono state svolte diverse prove inerenti a task differenti. Ogni scenario prevede l'uso di due veicoli reali a fronte di quattro agenti virtuali simulati al computer. Le ADF dello sciame sono:

- due gaussiane con FoV di $90°$, ampiezza $A = 2$ e dispersione $\sigma_x = \sigma_y = 2$;
- due gaussiane simmetriche con ampiezza $A = 2$ e dispersione $\sigma_x = \sigma_y = 1$.

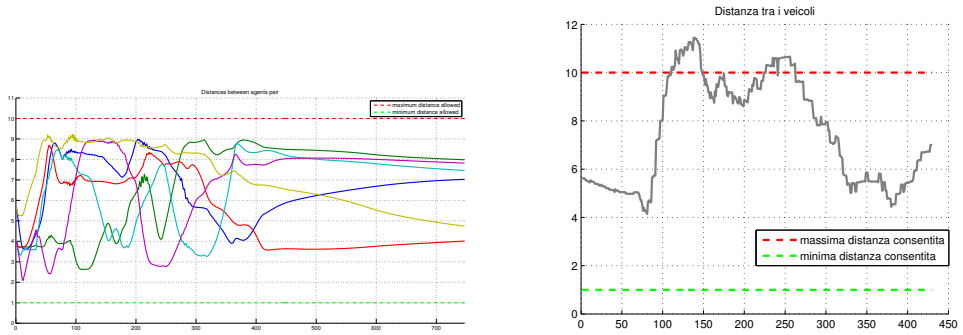
I veicoli reali inseguono la traiettoria generata dagli agenti con FoV. Gli stati iniziali sono rispettivamente $p_1 = [0, 0, -\frac{\pi}{4}]$, $p_2 = [4, 4, -\frac{\pi}{6}]$, $p_3 = [4, 0, \sim]$, $p_1 = [4, 0, \sim]$.

Nello scenario sono stati inseriti tre ostacoli di forme differenti ed è stata imposta un distanza minima da essi pari a $r = 1m$ e con distanza di rilevamento $R = 3m$. Inoltre sono stati aggiunti i vincoli di distanza interagente sviluppati nel Capitolo 3. La distanza minima fra agenti è $r = 1m$ mentre quella massima $d = 10m$. Le distanze di rilevamento per collision avoidance e connectivity maintenance sono rispettivamente $R = 4$ e $D = 7$ metri.



(a) **(b)**

Figura 7.13: Percorso eseguito dagli agenti simulati e da quelli reali.



(a) **(b)**

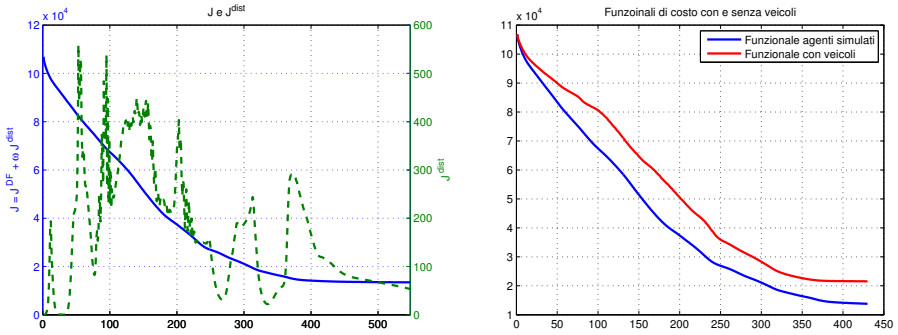
Figura 7.14: Distanze tra gli agenti virtuali e tra i due veicoli

7.8.1 Risultati dei test

Il primo test effettuato prevede che gli agenti svolgano un task di Effective Coverage. In Figura 7.13 è possibile osservare il percorso svolto dagli agenti (linee tratteggiate) e quello seguito invece dai veicoli reali (linee continue punteggiate). La "mini-car" segnata in nero realizza l'agente verde, mentre quella in grigio insegue l'agente rosso.

Si può notare che la posizione iniziale delle macchine non coincide con quella degli agenti. Questo avviene poiché anche lo stesso sistema di riferimento, utilizzato per passare da coordinate ECEF a NED, è affetto da errore in quanto ottenuto sempre attraverso un modulo GPS. Tuttavia in questo caso l'incertezza è inferiore a quella delle "mini-car" perché è stato possibile fare una media dei dati raccolti. Perciò, sebbene i veicoli fossero posizionati correttamente alla partenza, la somma tra errore di posizione di riferimento e quella del GPS presente sul veicolo, scostano di qualche metro la locazione calcolata dalle macchine.

In Figura 7.14a sono riportate le distanze tra gli agenti simulati, risultano ovviamente tutte all'interno dei bound previsti. Invece in Figura 7.14b è mostrata



(a) **(b)**

Figura 7.15: Funzionale J. **a:** funzionale ottenuto dagli agenti simulati. **b:** funzionale che si otterrebbe se gli agenti avessero seguito le traiettorie dei rispettivi veicoli.

la distanza fra i due veicoli. Si nota come essa sfori il limite superiore. Tuttavia il vincolo viene superato di appena due metri valore che rientra nell'incertezza di posizione dei GPS.

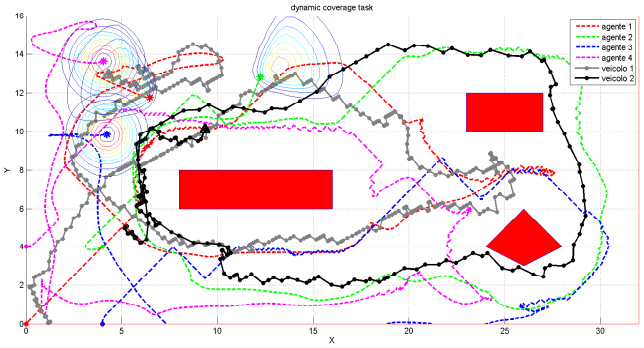
Infine in Figura 7.15 è possibile osservare l'andamento del funzionale di costo per il task di effective coverage. Nella prima immagine è rappresentato J ottenuto dagli agenti virtuali. Nella Figura 7.15b è invece rappresentato l'andamento di J^{DF} nel caso in cui gli agenti, a cui è associato un veicolo, avessero eseguito la traiettoria compiuta dalle "mini-car" e non quella della simulazione. Come ci si può immaginare il funzionale in quest'ultimo caso assume valori maggiori in quanto il percorso dei veicoli non è più ottimo per l'esecuzione del task in questione.

Dopo il task di effective coverage ne è stato svolto uno di dynamic coverage. I risultati sono tutti rappresentati in Figura 7.16. Sebbene il percorso svolto dagli agenti in Figura 7.16a risulti alquanto confusionario, si può notare come i vicoli non scontrino contro gli ostacoli e come seguano abbastanza fedelmente i relativi agenti.

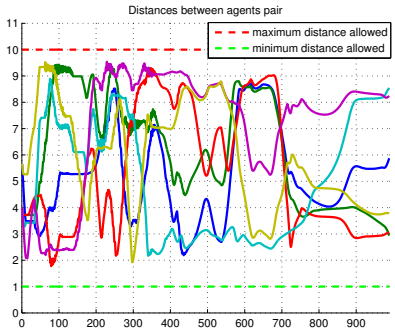
Anche in questo test le "mini-car" sforano i vincoli di distanze massime e minime. Tuttavia rispetto al caso precedente vengono superati solo per un attimo. Si ricorda inoltre che le posizioni e quindi le distanze sono affette dall'incertezza del GPS. Si può notare inoltre la somiglianza fra l'andamento della distanza fra i veicoli e quella dei rispettivi agenti in Figura 7.16b linea blu.

Infine si può osservare come, anche in questo caso, il funzionale J, che si sarebbe ottenuto se gli agenti con FoV avessero seguito il percorso dei veicoli reali, risulti più elevato di quello nominale. Tuttavia all'istante 430 i due funzionali risultano quasi coincidere.

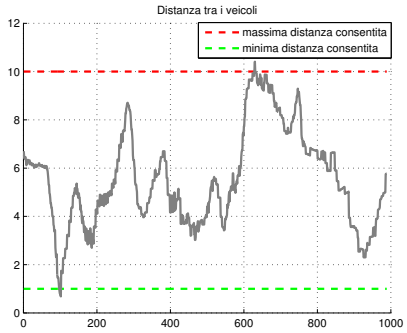
Infine si riportano i risultati ottenuti da un task di static coverage svolto però da un solo veicolo. L'agente, che genera la traiettoria inseguita dalla "mini-car" , è rappresentato dalla linea verde di Figura 7.17a. La TDF utilizzata è una gaussiana



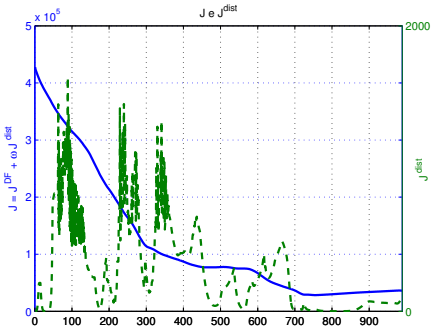
(a) *Percorso effettuato*



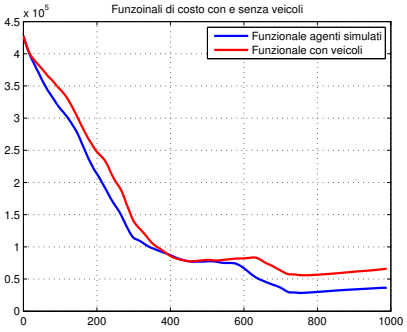
(b) *Distanze fra agenti simulati*



(c) *Distanze fra i veicoli*

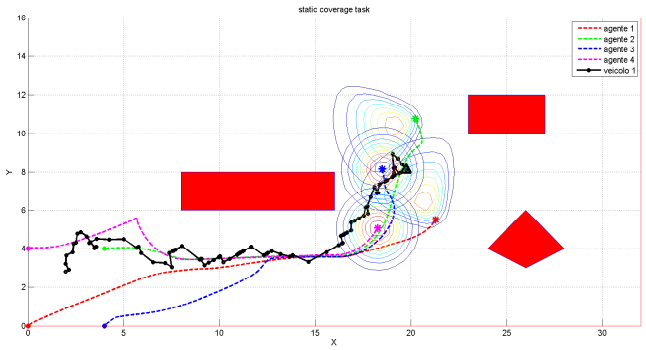


(d) *Funzionale con agenti simulati*

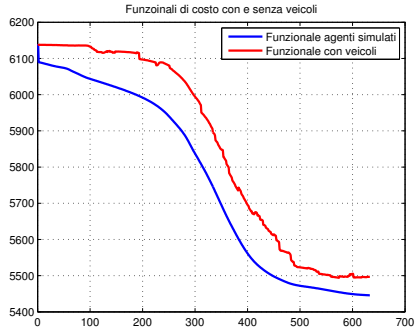


(e) *Funzionale con i due veicoli*

Figura 7.16: Risultati del test con Dynamic Coverage Task



(a) *Percorso effettuato*



(b) *J con e senza il veicolo*

Figura 7.17: Risultati del test con Static Coverage Task

centrata in $p_{TDF} = [18, 8]$. Il veicolo in questo caso passa molto vicino all'ostacolo, arrivando ad una distanza minima pari a $1.11m$ a fronte di un limite inferiore di un metro.

In questo caso l'andamento del funzionale in Figura 7.17b è molto frastagliato e a tratti risulta anche aumentare. Ciò è dovuto all'incertezza di posizione del GPS. Può accadere infatti che la posizione $p[n+1]$ risulti più distante dal target rispetto a $p[n]$ proprio perché l'errore ha "mosso" l'agente in una posizione più arretrata della precedente. Un altro motivo è dovuto al fatto che, in questo task, gli agenti virtuali si muovono più lentamente. Questo implica che il veicolo resta fermo per qualche secondo prima di inseguire nuovamente l'agente e ciò comporta che J resti costante per poi diminuire repentinamente.

7.9 Conclusioni

In questo capitolo è stato descritto il sistema di testing del DF Framework. Esso è composto da veicoli autonomi capaci di muoversi verso un waypoint, ossia la posizione

dell'agente virtuale simulato al pc. È stato descritto nel dettaglio l'hardware che compone le "mini-car" e il funzionamento dell'autopilota.

Di vitale importanza è inoltre l'infrastruttura di comunicazione sviluppata che si basa sul segnale PPS fornito dai ricevitori GPS. Sono stati svolti numerosi test con l'obbiettivo di verificare l'effettiva sincronizzazione e precisione di questo segnale.

Infine sono stati condotti i test. Le traiettorie generate dai veicoli risultano abbastanza vicine a quelle degli agenti simulati. La differenza fra questi due percorsi è dovuto in parte alla dinamica vincolata delle macchine, molto diversa dal singolo integratore con cui sono modellati gli agenti. Principalmente però l'impatto maggiore è dato dall'incertezza del posizionamento GPS. Si consideri tuttavia che se gli esperimenti fossero stati svolti su una superficie maggiore, l'errore del GPS avrebbe influito meno sui risultati finali.

Capitolo 8

Conclusioni e possibili sviluppi

Durante lo svolgimento del lavoro di tesi è stato analizzato, ampliato e perfezionato il Framework delle Funzioni Descrittive che permette di coordinare il moto di uno sciame di agenti per l'esecuzione di uno specifico task.

Inizialmente è stato mostrato come il Framework rappresenti gli agenti, e il task che deve essere portato a termine, sotto forma di funzioni dello spazio. In seguito è stato dimostrato rigorosamente, seguendo [Niccolini, 2011], come il controllo a discesa del gradiente permetta di minimizzare il funzionale di costo, associato alle performance dello sciame, anche in caso di task dinamici. Si è quindi discusso su quale possa essere il significato fisico del processo di ottimizzazione svolto dagli agenti.

In seguito il Framework è stato ampliato così da poter rappresentare eventuali ostacoli presenti nell'area di lavoro. A seguito di questa modifica si è reso necessario dimostrare nuovamente la convergenza e la stabilità delle legge di controllo. Tuttavia tale dimostrazione non assicura l'obstacle avoidance, sebbene gli ostacoli abbiano, di base, un effetto repulsivo sugli agenti. Per garantire che non avvengano collisioni è stata introdotta una dinamica che modifica le Funzioni Descrittive degli ostacoli in base alla posizione degli agenti. Anche in questo caso, si è discusso sulla possibile interpretazione della legge di controllo che muove lo sciame verso il task evitando gli ostacoli.

Dopo questa prima modifica, sono stati introdotti dei vincoli sulle distanze tra coppie di agenti. Questo significa che sono state aggiunte le proprietà di collision avoidance fra agenti, e connectivty maintenance dello sciame. Per fare ciò, è stato introdotto un nuovo funzionale di costo da affiancare a quello utilizzato dal DF Framework. Mantenendo quindi un approccio a discesa del gradiente è stata ottenuta una legge di controllo che vede gli agenti soggetti a due ingressi distinti: uno generato dal Framework e uno che assicura che i vincoli non vengano violati. Infine è stato

mostrato come l'aggiunta di queste proprietà, e in particolare della connettività dello sciame, porti in generale a risultati migliori nel task di effective coverage.

La connectivity maintenance è stata realizzata anche all'interno del DF Framework. Aggiungendo infatti una Funzione Descrittiva, detta di connettività, sul baricentro dello sciame si va a generare una componente attrattiva all'interno del controllo che muove gli agenti. Il risultato ottenuto porta lo sciame a spostarsi in maniera compatta. Anche in questo caso è stato necessario aggiungere una dinamica alla Funzione di Connettività per poter garantire che gli agenti non superassero una distanza massima prefissata dal loro centro di massa.

Aggiunte quindi le proprietà di obstacle/collision avoidance e connectivity maintenance si è rivolta l'attenzione verso la dinamica degli agenti. Questi infatti sono sempre stati considerati singoli integratori, come avviene in vasta parte della letteratura sugli sciami. Nel Capitolo 6 è stato invece dimostrato come il DF Framework possa lavorare anche con modelli di agenti del secondo ordine. Ovviamente per realizzare questa modifica è stato necessario cambiare la legge di controllo che ora non risulta più un metodo del gradiente ma un metodo di ottimizzazione *quasi-Newton*. Sono state infine confrontate le due dinamiche attraverso esempi e simulazioni.

Il lavoro teorico svolto utilizza costantemente un approccio di tipo centralizzato: ogni agente conosce tutto degli altri elementi dello sciame. Sarebbe interessante sviluppare in futuro un approccio di tipo decentralizzato in cui ogni agente possiede informazioni solo su un numero ristretto di elementi attorno a lui. Dando per scontata la forma della ADF, agli agenti basterebbe conoscere la posizione e l'orientazione di tutti gli altri per poter calcolare correttamente il controllo (anche con la presenza di ostacoli o di vincoli). Nel caso in cui fossero note le informazioni di una parte dello sciame, gli agenti avrebbero una stima del funzionale di costo sicuramente superiore (o uguale) a quella del funzionale reale. Si pensi ad esempio ad un task di effective coverage: l'agente i-esimo potrebbe passare in aree già perlustrate da altri agenti senza esserne a conoscenza.

Ulteriori sviluppi del DF Framework potrebbero interessare l'analisi dei punti critici del funzionale di costo J. Infatti uno dei problemi principali vede lo sciame fermarsi su minimi locali anziché su quello globale. Forse sviluppando nuove funzioni di costo $f(\cdot)$ si potrebbe garantire la presenza di un solo minimo all'interno del funzionale e di conseguenza garantire sempre il corretto svolgimento del task.

Al fine di validare il lavoro teorico appena trattato, è stato messo appunto uno hardware testbed. Sono stati quindi sviluppati dei veicoli autonomi che inseguono la traiettoria generata dagli agenti simulati al pc. In questo modo è stato possibile osservare quali risultati può fornire il DF Framework se applicato ad un sistema reale e con dinamica vincolata, quale quello dei veicoli terrestri. Il testbed, e l'infrastruttura di comunicazione che lo supporta, sono stati realizzati in maniera scalabile così da

permettere l'aggiunta si nuovi veicoli ed essere utilizzato anche per altri fini.

Sviluppi futuri potrebbero chiudere l'anello che lega agenti simulati e veicoli reali. Infatti nell'applicazione svolta, i veicoli inseguono gli agenti virtuali come fossero waypoint ma non avviene il viceversa. Gli agenti non risentono affatto del moto dei veicoli reali. Per chiudere l'anello in feedback, sarebbe necessario aggiungere un filtro di stima sulla posizione degli agenti in quanto troppo affetta da rumore. Per realizzare tale filtro sarebbe inoltre opportuno dotare i veicoli di un sensore inerziale, così da poter svolgere un processo di *sensor fusion* fra GPS e IMU.

Bibliografia

Bertsekas, Dimitri P

1999 «Nonlinear programming». (Citato alle p. 42, 75.)

Bonabeau, Eric, Marco Dorigo e Guy Theraulaz

1999 *Swarm intelligence: from natural to artificial systems*, 1, Oxford university press. (Citato a p. xi.)

Bryson, Arthur E, Sheldon Baron e Yu-Chi Ho

1965 «Differential games and optimal pursuit-evasion strategies», in *Joint Automatic Control Conference*, 3, p. 37-40. (Citato a p. 61.)

Bullo, Francesco, Shaunak D Bopardikar e Joao P Hespanha

2007 «Cooperative pursuit with sensing limitations», in *American Control Conference, 2007. ACC'07*, IEEE, p. 5394-5399. (Citato a p. 61.)

Bullo, Francesco, Jorge Cortés e Sonia Martinez

2009 *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms: A Mathematical Approach to Motion Coordination Algorithms*, Princeton University Press. (Citato alle p. xi, 10, 11, 71.)

Chang, Dong Eui, Shawn C Shadden, Jerrold E Marsden e Reza Olfati-Saber

2003 «Collision avoidance for multiple agent systems». (Citato a p. 22.)

Della Santina, Cosimo e Aringhieri Stefano

2011 «Sviluppo di un Testbed per l'analisi di algoritmi di cooperazione fra veicoli». (Citato a p. 91.)

Ferrari Braga, Andrea

2013 «Controllo di uno Sciame di Robot utilizzando il Framework delle Funzioni Descrittive». (Citato alle p. 1, 6, 8, 13, 34, 50.)

Ferrari Braga, Andrea, Mario Innocenti e Lorenzo Pollini

s. d. «Multi-Agent Coordination with arbitrarily shaped Descriptor Functions». (Citato alle p. xii, 1, 6, 7, 21, 40.)

Foster, John

2011 *Xbee cookbook issue 1.4 for series 1 (freescale) with 802.15. 4 firmware.* (Citato a p. 85.)

Howard, Andrew, Maja J Matarić e Gaurav S Sukhatme

2002 «Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem», in *Distributed Autonomous Robotic Systems 5*, Springer, p. 299-308. (Citato a p. 10.)

Hussein, Islam, Dušan M Stipanović et al.

2006 «Effective coverage control using dynamic sensor networks», in *Decision and Control, 2006 45th IEEE Conference on*, IEEE, p. 2747-2752. (Citato alle p. 3, 11.)

2007a «Effective coverage control for mobile sensor networks with guaranteed collision avoidance», *Control Systems Technology, IEEE Transactions on*, 15, 4, p. 642-657. (Citato alle p. xi, 36, 40, 42.)

2007b «Effective coverage control using dynamic sensor networks with flocking and guaranteed collision avoidance», in *American Control Conference, 2007. ACC'07*, IEEE, p. 3420-3425. (Citato a p. 42.)

Hussein, Islam, Dušan M Stipanović, Yue Wang et al.

2007 «Reliable coverage control using heterogeneous vehicles», in *Decision and Control, 2007 46th IEEE Conference on*, IEEE, p. 6142-6147. (Citato alle p. 11, 42.)

Isaacs, Rufus

1954 «Differential games». (Citato a p. 61.)

Kennedy, James, James F Kennedy, Russell C Eberhart e Yuhui Shi

2001 *Swarm intelligence*, Morgan Kaufmann. (Citato a p. xi.)

LaValle, Steven M

2006 *Planning algorithms*, Cambridge university press. (Citato alle p. xi, 22, 42, 61, 71.)

Lynch, Nancy A

1996 *Distributed algorithms*, Morgan Kaufmann. (Citato a p. 71.)

Masoud, Ahmad, Samer Masoud et al.

2000 «Motion planning in the presence of directional and obstacle avoidance constraints using nonlinear, anisotropic, harmonic potential fields», in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, IEEE, vol. 3, p. 2944-2951. (Citato a p. 22.)

Meguerdichian, Seapahn, Farinaz Koushanfar, Gang Qu e Miodrag Potkonjak

2001 «Exposure in wireless ad-hoc sensor networks», in *Proceedings of the 7th annual international conference on Mobile computing and networking*, ACM, p. 139-150. (Citato a p. 10.)

Merayo, José MG, Peter Brauer, Fritz Primdahl, Jan Raagaard Petersen e Otto V Nielsen

2000 «Scalar calibration of vector magnetometers», *Measurement science and technology*, 11, 2, p. 120. (Citato a p. 85.)

Mesbahi, Mehran e Magnus Egerstedt

2010 *Graph theoretic methods in multiagent networks*, Princeton University Press. (Citato alle p. xi, 71.)

Murphy, Robin R

2000 *Introduction to AI Robotics*, The MIT press. (Citato a p. 22.)

Niccolini, Marta

2011 «Swarm Abstractions for Distributed Estimation and Control». (Citato alle p. 1, 3, 14, 16, 17, 19, 21, 58, 98.)

Niccolini, Marta, Mario Innocenti e Lorenzo Pollini

2010 «Near optimal swarm deployment using descriptor functions», in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, p. 4952-4957. (Citato alle p. xii, 1, 2.)

Niccolini, Marta, Lorenzo Pollini e Mario Innocenti

2014 «Cooperative control for multiple autonomous vehicles using descriptor functions», *Journal of Sensor and Actuator Networks*, 3, 1, p. 26-43. (Citato a p. 1.)

Pallottino, Lucia, Andrea Caiti, Simone Nardi e Cosimo Della Santina

2015 «A game theoretic approach for the coordination of underwater autonomous robots against asymmetric threats». (Citato a p. 59.)

Pallottino, Lucia, Simone Nardi e Cosimo Della Santina

2014 «Coverage in dynamic environment: a game theoretic approach». (Citato a p. 59.)

Passino, Kevin M

2005 *Biomimicry for optimization, control, and automation*, Springer Science & Business Media. (Citato a p. xi.)

Sastry, Shankar, Adonis Antoniades e H Jin Kim

2003 «Pursuit-evasion strategies for teams of multiple agents with incomplete information», in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, IEEE, vol. 1, p. 756-761. (Citato alle p. 60, 65.)

Sastry, Shankar, H Jin Kim, Rene Vidal, David H Shim e Omid Shakernia

2001 «A hierarchical approach to probabilistic pursuit-evasion games with unmanned ground and aerial vehicles», in *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, IEEE, vol. 1, p. 634-639. (Citato alle p. 61, 65.)

Schwager, Mac, Jean-Jacques Slotine e Daniela Rus

2007 «Decentralized, adaptive control for coverage with networked robots», in *Robotics and Automation, 2007 IEEE International Conference on*, IEEE, p. 3289-3294. (Citato a p. 11.)

Sharifi, Farid, Abbas Chamseddine, Hamid Mahboubi, Youmin Zhang e Amir G Aghdam

2015 «A distributed deployment strategy for a network of cooperative autonomous vehicles», *Control Systems Technology, IEEE Transactions on*, 23, 2, p. 737-745. (Citato alle p. 11, 72, 73.)

Stipanović, Dušan M, Peter F Hokayem, Mark W Spong e Dragoslav D Šiljak

2007 «Cooperative avoidance control for multiagent systems», *Journal of Dynamic Systems, Measurement, and Control*, 129, 5, p. 699-707. (Citato a p. 40.)